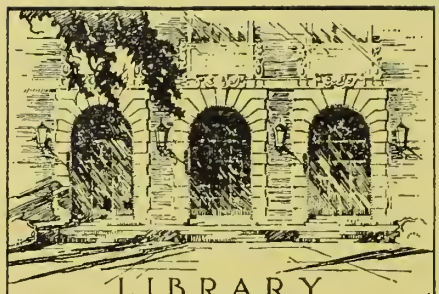


Report No. 240

COO-1469-0068

A GRAPHICS DISPLAY LANGUAGE

by

Dugald Austin Henderson, Jr.

August 21, 1967

DEPARTMENT OF COMPUTER SCIENCE · UNIVERSITY OF ILLINOIS · URBANA, ILLINOIS

Report No. 240

A GRAPHICS DISPLAY LANGUAGE*

by

Dugald Austin Henderson, Jr.

August 21, 1967

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

* This work was supported in part by Contract No. US AEC AT(11-1)1469 and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, August, 1967.

ACKNOWLEDGMENT

The author wishes to acknowledge the invaluable guidance and encouragement given to him by Dr. C. W. Gear during all phases of this study. Also thanks are extended to James Roman for his extensive work in programming many of the geometrical routines, Michelle Boucher who typed the rough draft, and Marguerite Dunlap who typed the final draft.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

BCU - basic construction unit

BCUC - basic construction unit command

BCUL - basic construction unit line

FL - figure line

NPI - no pointer indicator

NSI - next sequential instruction

PAM - parameter and manipulation block

SF - subfigure

SFC - subfigure command

SFL - subfigure line

I. MOTIVATION AND INTRODUCTION

It has become clear that one of the most powerful roles the computer will play in the future is as a partner to a human being in an on-line exchange type of creative process. The computer will act as the powerful computational half of the man-machine pair, leaving the human to provide creative insight, and direction.

To make such facilities for computer-aided thinking available to those unfamiliar with these devices, it is essential to provide software packages that allow the communication between man and machine to be in a form familiar to the human participant.

In the design of systems allowing the creation of two-dimensional pictures the use of a Cathode Ray Tube (CRT) display screen, light-pen, and special function keys have been the usual medium of communication. "Sketchpad" was the first of the devices to use this method.[1]

The CRT display screen has become a very common device for many applications. However, special light-pen and function-keys are not so common; this follows from there being a large number of applications requiring displayed output but typed input. Many business applications are of this sort.

The relatively ready availability of installations with display screens - console typewriter peripheral units leads to the desirability of a language through which the operations of the graphical displays of the above-mentioned systems could be created by keying in information. It is desirable to provide the language of basic Euclidean geometry as there is widespread understanding and acceptance

of this description of displays.

Many installations lacking CRT displays have off-line plotters readily available. These devices would prove useful drafting tools provided that one could easily describe figures to be drawn by them. A language of the type proposed above would be such a description. The fast interactive abilities of the on-line CRT display would not carry over to the system having only plotter output; however, most programs are written in the slow interactive environment of batch processing, a very tried and proven milieu. In short, a computer with a Fortran compiler and a plotter should provide, through use of the language proposed here, a very powerful drafting tool.

The software package described in this paper is a preliminary attempt to provide such a language. This Graphics Display Language (G.D.L.) provides the facility to describe in a fairly rigid format, and have displayed, figures composed of straight line segments and arcs of circles. The choice of such an apparently restricted language was made to limit the scope of the work, so as to direct study toward data structure and implementation. As a result the restrictions can be easily relaxed. The data structures used here can, without modification, be employed to encompass the drawing of conic sections and more complex curves. A more sophisticated string processing segment can be added to allow a freer format.

The aim of this work was not to provide a finished software system but rather to provide a basic package which can be used to study the worth of this approach to computer graphics; it should then provide a base for further development. Hence, thought has been put into the

facilities which may be desired and the data structures designed to be amenable to as much flexibility as possible.

Little is known of what functional commands it is desirable to give a designer using such a system. Initial assumptions as to what will be most useful have had to be made. It is anticipated that usage of the package will accent the weakness in these assumptions, and the G.D.L. amended to correct them.

II. THE CHOICE BETWEEN INTERPRETATION AND COMPILATION

As any drawing can be described as a list of line segments, arcs, and points (one may or may not want to draw this last), an initial reaction is to do all calculations of a new component from data for existing components at the time it is entered, and store the name and the values of the co-ordinates only. Lists of components would thus be formed; description of any component would be in terms of actual co-ordinates or components already defined. To draw the figure, one has a subroutine to display each basic type of component, which one then does for each member of each list.

However, the advantage of on-line operation is the ability to change figures one has drawn. Suppose calculation of a point C as the intersection of lines A and B has taken place and its co-ordinates stored in the list of points. Now the definition of line B is changed. It must be possible to recalculate C. This involves knowing which things depend on what other things, and also the nature of these dependences.

If all of this information must be stored, one might as well store all the input statements and provide an interpretive system which would analyze and draw any file of statements in the language.

However, the input statements are in a language particularly well suited to the human using the system and hence not well suited to easy or economical storage within the computer.

Other advantages and disadvantages of these two extremes of compilation and interpretation are:

a) Compilation provides faster display time, all the work having been done at list-creation time.

b) Storage space for the interpretive system will in general be larger.

c) Less computer time will be taken in the interpretive system, as the programs to accept each line are the standard file generation routines of the time-sharing system. In the compilation system, the compiler must be called for each line.

d) The interpretive system has flexibility allowing changing, adding and deleting, while the compiler has no ability this way.

Between these extremes, there are many levels of partial interpretation. These involve partially analyzing the input statements, and storing the information contained therein in a partially digested fashion in a format designed to be economical on storage. The level of digestion determines the degree of flexibility obtained. The more digested, the harder it is to change things. Specifically, if interrelational information is used at input-time, the retention of it is redundant; yet this retention is the only possible way to insure flexibility.

The scheme selected initially (GDL-1) was one in which relational information was retained in the form of pointers, and names were compiled.

The second scheme (GDL-2) is much more interpretive, call-by-name being used throughout.

III. THE LANGUAGE

a) Definition

The GDL is divided into two sections. The first of these includes the statements which define the figures to be drawn. These statements are given in the context of the commands of the second section called the HANDLER. These commands are instructions to the GDL compiler providing information necessary to properly compile the definition statements. The Handler commands are different for GDL-1 and GDL-2 and hence will be dealt with in the description of each of these compilers, its data structures, capabilities, and uses. In this section a discussion will be given of the definition statements. Once these are understood, the use of the language will become apparent, and the discussion of the Handlers will follow naturally.

Figures described by GDL are composed of segments of straight lines and circles, and subfigures previously described. The description is given in a series of 'lines' of character-string input. Each line may physically spill over onto more than one card or extend onto a second line of typed input on a console. At the current stage of GDL the format of these lines has been rigidly fixed to allow a minimum of effort to be spent on parsing of the input string. It is expected that when more experience has been acquired in using GDL, the desirable condensing of statements will become clear. At that time free format and these condensations should be added.

The format is given in Figure 1.

First a line with key word FIGURE (this will be called a FIGURE-line) is given. The name of the figure is given in the name word. All other words are blank.

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 58--63 65--70 72

DRAW NAME-- KEY---  N WORD-1 WORD-2 WORD-3 WORD-4 WORD-5 WORD-6 WORD-7 1
                      WORD-8 WORD-9 WORD10 WORD11 WORD12 WORD13 WORD14 1
                      WORD15 WORD16 WORD17 WORD18 WORD19 WORD20 WORD21 1
                      WORD22 WORD23 WORD24 WORD25
```

Figure 1. GDL Line Format

All names are character strings of six alphabetic and numeric characters (no special characters) with the first character an alphabetic character. Blanks are allowed at the right-hand end of a name. (The name ABC ␣␣␣ will be shown ABC here.) Following the FIGURE-line, the description of the basic construction units (BCU's) are given. These may be points, line segments, circles, arcs, distances, or angles. Following the common practice in Euclidean geometry, the figure to be described is augmented with 'construction lines' to make it easier to describe or draw. These are used to construct the figure but are not drawn when the figure is drawn, (physically the construction lines are erased once the figure is drawn). If a BCU is to be drawn as part of the figure the draw sord is DRAW. If this is blank the BCU is taken to be a construction line only. (Note that anything not drawn is considered a construction line in our case. Thus, most points, and all distances and angles are 'construction lines').

Each BCU, drawn or not, is given a name. In GDL-1 these names must be unique for all figures. In GDL-2 names must be unique within each figure.

The integer N gives the kind of BCU of the type indicated by the KEY word that is being described. For example, a point can be described by its co-ordinates (kind 1), or by the intersection of two line segments (kind 2), or the intersection of a line and an arc (kins 6), etc. Each kind of BCU of a given type is numbered arbitrarily. A complete list of those implemented in GDL-1 and GDL-2 is given at the end of this section. As will be noted in this list, many of the BCU's have quite arbitrary definitions. It is obvious that many useful BCU's

are not yet implemented, and probable that some BCU's in the current list are quite useless. Again, the current work is to provide some study chiefly of the data structures for GDL. It is anticipated that given this, useful BCU's can be easily added to the list. The structures provided by both GDL-1 and GDL-2 are open-ended in this respect for exactly this reason.

Currently, the BCU's implemented are of the following types: POINT ␣ , LINE ␣␣ , CIRCLE, ARC ␣␣␣ , DIST ␣␣ , ANGLE ␣ .

The word 1 through word 25 positions are used to give the BCU's and values in terms of which the BCU in the current line is being described. For example if the point A (4, 3) is to be defined by its co-ordinates, word 1 is given value 4 and word 2 the value 3. If two points A and B are defined, then the line segment AB is of kind 1 with word 1 given the name A ␣␣␣␣␣ and word 2 the name B ␣␣␣␣␣ .

Values are six numeric characters and blanks. Blanks are interpreted as zeros. The decimal point is assumed to be between the third and fourth digit. Digit 1 set - gives a negative value.

An an example consider the figure described by the GDL - lines of Figure 2. This is a square with 8 units to each side. A point of kind 1 requires two co-ordinates to define it. A line of kind one is defined as connecting the two points given.

An iscoceles triangle is described by the lines of Figure 3.

Once a complete figure is defined, it may be called as part of a larger figure.

The KEY - word is SUBFIG on such a subfigure call. Word 1 is the name of the figure being called as a subfigure. Name is the

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

     SQUARE FIGURE
     SQUA   POINT   1   0       0
     SQUB   POINT   1   0       8
     SQUC   POINT   1   8       8
     SQUD   POINT   1   8       0
DRAW SQUAB  LINE    1 SQUA    SQUB
DRAW SQUBC  LINE    1 SQUB    SQUC
DRAW SQUCD  LINE    1 SQUC    SQUD
DRAW SQUAD  LINE    1 SQUA    SQUD
```

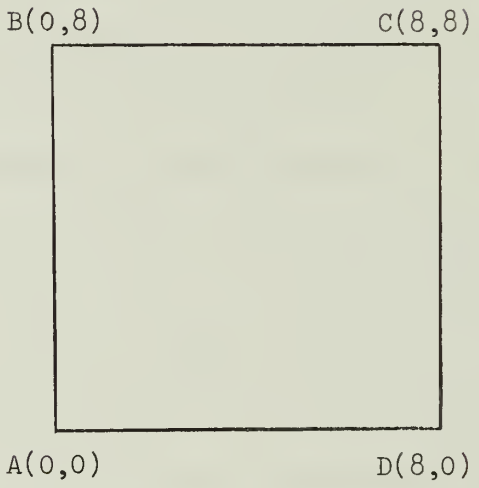


Figure 2. A Fixed Square

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

     TRIANG FIGURE
     TRIA   POINT   1   0       0
     TRIB   POINT   1   4       6
     TRIC   POINT   1   8       0
DRAW TRIAB  LINE    1 TRIA    TRIB
DRAW TRIBC  LINE    1 TRIB    TRIC
DRAW TRICA  LINE    1 TRIC    TRIA
```

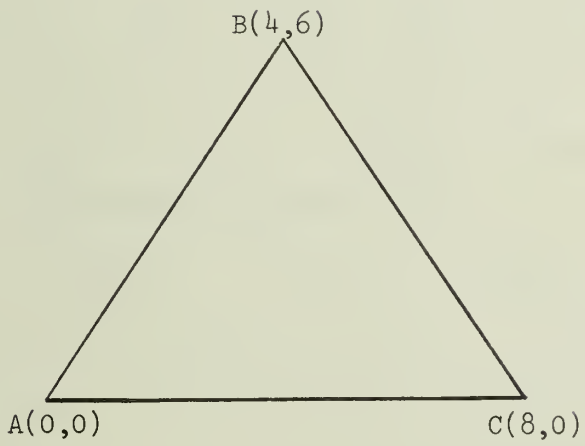


Figure 3. A Fixed Triangle

subfigure as part of the current figure. (Name serves no real purpose in the existing GDL's, but an important extension of the language should allow indications of the type ABC·DEF·G where G is a BCU of some figure which is given the name DEF as a subfigure of some figure which is given the name ABC as a subfigure of the current figure.) N is not used.

Figure 4 presents a figure defined as a composite of our previously defined figures SQUARE and TRIANGLE. (It should be noted that this is purely an illustrative example and hence is missing many necessary words of definition.) Also note that all subfigures are assumed not to be construction lines. Hence, the DRAW keys are unnecessary.

This example indicates a definite further need. In calling a subfigure, one may well wish to have its origin translated with respect to the origin of the current figure. Also one may wish to rotate and/or expand the subfigure about its origin for use in the current figure. Words 2 through 5 are used for these values. Word 2 = x_0, word 3 = y_0, word 4 = θ, word 5 = E, where (x_0, y_0) is the translation, θ is the angle of rotation in degrees, and E is the expansion factor.

The composite of Figure 5 is given by the same GDL - lines as Figure 4 with the necessary manipulations. Note that the square has not been manipulated at all (translation by (0, 0), 0° rotation, expansion factor of 1).

In describing a figure, one may wish to describe many similar figures by providing parameters. When a figure is called as a subfigure, if parameters have been used in its description, these must

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

     COMP1  FIGURE
DRAW C1FIG1 SUBFIG    SQUARE
DRAW C1FIG2 SUBFIG    TRIANG
```

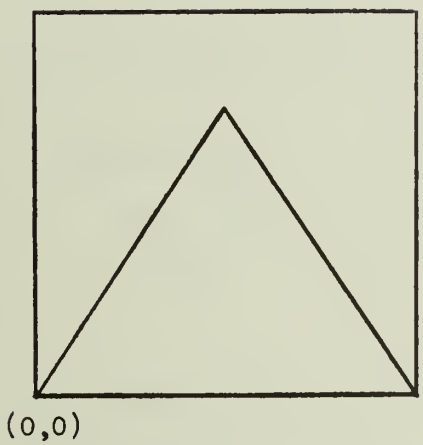


Figure 4. A Figure Composed of Unmanipulated Subfigures

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

     COMP2  FIGURE
     C2FIG1 SUBFIG     SQUARE   0       0       0       1
     C2FIG2 SUBFIG     TRIANG   6       6     180       05
```

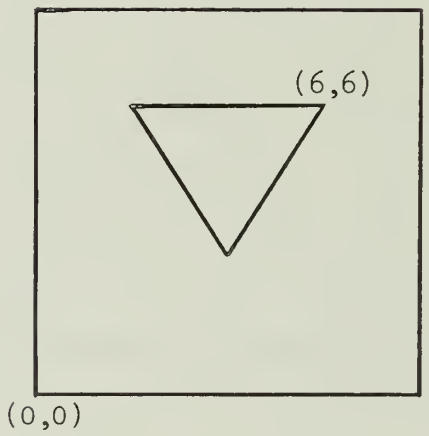


Figure 5. A Figure Composed of Manipulated Subfigures

be provided. Up to 20 parameters (numbered 01 through 20) may be used in the description of a figure. Suppose a point C is described in terms of its co-ordinates (x, 3). If parameter 4 is the one to be used to deliver the value x to this figure, word 1 of the GDL - line describing C would be PAR␣04. (Note that as internal blanks are not allowed in names this will not be mistaken for a name.) Words 6 through 25 of the SUBFIG - line are used to provide the parameters to the figure being called as a subfigure. Each parameter in the call may be a value, a name of the BCU of the correct type, or a parameter itself. Hence, a parameter can be passed down through nested subfigures (as in Fortran). A SUBFIG - line need not have all 25 words present. Only the necessary ones (maybe only words 1 through 5) need be provided.

An example of a more general figure is given in Figure 6. Part a) shows a rectangle with sides given by parameters 1 and 2. Part b) gives the GDL - line calling for an unmanipulated rectangle with sides 3 units and the distance given in the BCU described by GDL - line XDIST. (Note that while a point of kind 1 is expecting two values, it will search out and calculate a distance in place of that value. The opposite is also useful, and has been implemented. If in looking for a distance the program encounters a value in its search it will be satisfied.)

The current GDL's require that names used in lines be already defined (as BCU's or as figures as the case may be). This 'pre-definition' was not considered a serious restriction on the user, and as it simplifies the implementation it was employed.

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 58--63 65--70 72

     RECTAN FIGURE
     RECA   POINT   1   0       0
     RECB   POINT   1   0     PAR C2
     RECC   POINT   1 PAR 01 PAR C2
     RECD   POINT   1 PAR 01   0
DRAW RECAB  LINE    1 RECA    RECB
DRAW RECBC  LINE    1 RECB    RECC
DRAW RECCD  LINE    1 RECC    RECD
DRAW RECDA  LINE    1 RECD    RECA
             .
     USER   FIGURE
             .
     XDIST  DIST    X XXXXXX XXXXXX
             .
     XSUB   SUBFIG    RECTAN   0      0      0      1      3    XDIST
             .
```

Figure 6. The Use of Parameters

b) Basic Construction Units

The preceding discussion of the drawing program shows that the coding to analyze each type and kind of BCU is essentially a node in the tree describing the figure to be drawn. Each routine calls other routines "below" it in the tree; these calls cause values to be returned through the accumulator to the routine. The routine uses these values to calculate values describing the BCU it represents. If the routine is called at the first level below the figure, the routine returns the values to the plotting or displaying routines. If the routine is called at a lower level, the values calculated are returned through the accumulator to the calling routine.

The currently implemented routines are:

1100: a point defined by to co-ordinates

format: NAME PØINT 1 value 1 value 2

value 1 is the x co-ordinate

value 2 is the y co-ordinate

1200: a point defined by the intersection of two lines

format: NAME PØINT 2 LINE1 LINE2

description: the intersection of the line segments LINE1 and LINE2 extended if necessary

errors: LINE1 parallel to LINE2

magnitude of values of the co-ordinates of the intersection greater than 30.000

1300: a point defined by a translation from a point

format: NAME PØINT 3 PØINT1 value 1 value 2

1400: center of a circle

format: NAME PØINT 4 CIRCLE

2100: a line segment defined by two end points

format: NAME LINE 1 PØINT1 PØINT2

2200: a line segment defined by a point and a translation

format: NAME LINE 2 PØINT1 value 1 value 2

4100: a circle defined by center and radius

format: NAME CIRCLE 1 PØINT1 DIST1

4200: a circle defined by center and a point on the circumference

format: NAME CIRCLE 2 PØINT1 PØINT2

description: PØINT1 is the center

PØINT2 is a point on the circumference

errors: radius less than 0.020

4300: a circle defined by three points on the circumference

format: NAME CIRCLE 3 PØINT1 PØINT2 PØINT3

errors: points co-linear

5100: an arc defined by a circle and two angles

format: NAME ARC 1 CIRCLE ANG1 ANG2

description: the angles are measured counter-clockwise from positive horizontal radius; arc is drawn from first to second angle along circumference of the circle in the counter-clockwise direction

errors: ANGL equals ANG2

5200: an arc defined by a circle and two points

format: NAME ARC 2 CIRCLE PØINT1 PØINT2

description: an arc is drawn in the counter-clockwise direction from the radius vector on which or on whose extension lies PØINT1 to the similar radius vector defined by PØINT2

5300: an arc defined by a circle, a point and an angle

format: NAME ARC 3 CIRCLE PØINT1 ANG1

description: POINT1 defines one end of the arc (see 5200); the other end is at ANG1 from this. ANG1 may be negative in which case the arc is drawn clockwise through as many radians.

5400: an arc determined by a circle, a point and a distance

format: NAME ARC 4 CIRCLE PØINT1 DIST1

description: PØINT1 defines one end of the arc (see 5200): the other end is such that the chord joining the ends is DIST1 long. If DIST1 is negative the arc is drawn clockwise from the first point.

errors: DIST1 greater than diameter of circle

DIST1 zero

5500: an arc defined by a circle, an angle and a distance

format: NAME ARC 5 CIRCLE ANG1 DIST1

description: one end point of the arc is determined as ANG1 radians counter-clockwise from the positive horizontal radius; DIST1 is the chord distance to the other end point (see 5400)

5600: an arc defined by a circle and two distances

format: NAME ARC 6 CIRCLE DIST1 DIST2

description: DIST1 is the chord length to the first end point from (r,0) where r is the radius of the circle. The second end point is determined similarly by DIST2. The arc is drawn counter-clockwise from the first to the second end point.

6100: the distance between two points

format: NAME DIST 1 PØINT1 PØINT2

6200: horizontal distance between two points

format: NAME DIST 2 PØINT1 PØINT2

description: distance between base of perpendicular to the horizontal asix from POINT1 and POINT2. This is always positive

6300: vertical distance between two points

format: NAME DIST 3 PØINT1 PØINT2

description: distance between base of perpendicular to the vertical axis from PØINT1 and PØINT2. This is always positive

6400: perpendicular distance from a point to a line

format: NAME DIST 4 PØINT1 LINE1

6500: radius of a circle

format: NAME DIST 5 CIRCLE

6600: distance given by a real number

format: NAME DIST 6 value 1

6700: binary operations on distances

format: NAME DIST 7 DIST1 value 1 DIST2

description: binary operations on DIST1 and DIST2 are performed on the basis of the value of value 1

value 1 = 1 DIST1 + DIST2

2 DIST1 - DIST2

3 DIST1 x DIST2

4 DIST1 / DIST2

5 $DIST1^{DIST2}$ (exponentiation)

6 $DIST1 \sqrt{|DIST2|}$

7 DIST1 |DIST2|

7100: angle determined by two lines

format: NAME ANGLE 1 LINE1 LINE2

description: positive angle through which LINE1 must be rotated to get LINE2

7200: angle determined by three points

format: NAME ANGLE 2 PØINT1 PØINT2 PØINT3

description: positive angle through which the line defined by PØINT1 and PØINT2 must be rotated counter-clockwise to get the line defined by PØINT2 and PØINT3

7300: an angle given by a real number

format: NAME ANGLE 3 value 1

The program is designed to provide the user with a standard set of linkages to the Handler and Drawing programs for both GDL-1 and GDL-2. These can be used in creating new routines, if the following conventions are followed:

a) Communication in the accumulator:

Type	Description	MACC1	MACC2	MACC3	MACC4	MACC5
point	(x_1, y_1)	x_1 x 1000	y_1 x 1000			
line	(x_1, y_1) to (x_2, y_2)	x_1 x 1000	y_1 x 1000	x_2 x 1000	y_2 x 1000	
circle	center (x_1, y_1)	x_1 x 1000	y_1 x 1000	d x 1000		
arc	center (x, y_1)	x_1 x 1000	y_1 x 1000	d x 1000	α_1 x 1000	α_2 x 1000
	radius d					
	start angle α_1 $\lvert\alpha_1\rvert \leq \Pi$					
	end angle α $\lvert\alpha_2\rvert < \Pi$					
	drawn in counter-clockwise direction					
distance	d	d x 1000				
angle	α; $\lvert\alpha\rvert < 2\ \Pi$	α x 1000				

b)

Function to be Performed	Linkage in GDL-1	Linkage in GDL-2
1. Call a value routine	CALL IVAL (NSI + LT, NPAR) IF (NPAR) 5, 99, ℓ ℓ MACC1 = NPAR - 60000 LT = LT + 1	N = 8 K = IGET (NSI, LT, N) IF (K) ℓ, 9990, 9031 ℓ MACC1 = N LT = LT + 1
2. Call a distance routine	CALL IDIST (NSI + LT, NPAR) IF (NPAR) ℓ, 99, 4 ℓ MACC1 = -NPAR - 60000 LT = LT + 1	N = 6 K = IGGT (NSI, LT, N) IF (K) ℓ, 9990, 9031 ℓ MACC1 = N LT = LT + 1

3. Call a: point (n = 1) line (n = 2) circle (n = 4) arc (n = 5) angle (n = 7)	N = n GØ TØ 2	N = n GØ TØ 9030
4. Once the accumulator has been loaded with description of: point (n = 1) line (n = 2) circle (n = 4) arc (n = 5) dist (n = 6) degree (n = 7)	GØ TØ n050	GØ TØ n050

c) An Example

The lines of GDL given below describe the diagram of Figure 7.

Note that parameter 1 is used in the description of figure RECT to give the ratio of the lengths of the sides of the rectangle. Parameter 1 is also used in the description of figures LETG, LETD, and LETL to vary the shape of the letters described. Figure GDL comprising the three letters allows a variation in letter size in the figure by providing separate parameters for each letter. Similarly, the shapes of all four composing figures can be controlled individually. Figure 7 is just one of many diagrams possible to create by displaying the figure EXAMPL with various values for parameters 1 to 4.

Lines of GDL Describing Figure EXAMPL

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 58--63 65--70 72


     RECT   FIGURE
     RECT1  POINT   1 PAR 01   1
     RECT2  POINT   1 PAR 01 - 1
     RECTD1 DIST    7 - 1      3    PAR 01
     RECT3  POINT   1 RECTD1 - 1
     RECT4  POINT   1 RECTD1   1
DRAW RECT12 LINE    1 RECT1  RECT2
DRAW RECT23 LINE    1 RECT2  RECT3
DRAW RECT34 LINE    1 RECT3  RECT4
DRAW RECT41 LINE    1 RECT4  RECT1

     LETG   FIGURE
     LETGP1 POINT   1   0      0
     LETGC1 CIRCLE  1 LETGP1   1
DRAW LETGA1 ARC     1 LETGC1   0      3142
DRAW LETGL1 LINE    2 LETGP1   2      0
     LETGP2 POINT   1 - 1      0
DRAW LETGL2 LINE    2 LETGP2   0    PAR 01
     LETGP3 POINT   1   0    PAR C1
     LETGC2 CIRCLE  1 LETGP3   1
DRAW LETGA2 ARC     1 LETGC2   0      3142
```

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 58--63 65--70 72


     LETD   FIGURE
     LETDP1 POINT   1   0      0
     LETDC1 CIRCLE  1 LETDP1   1
DRAW LETDA1 ARC     1 LETDC1 - 1571   0
     LETDP2 POINT   1 - 1    - 1
     LETDP3 POINT   1   0    PAR C1
     LETDD1 DIST    7   1      1    PAR 01
     LETDP4 POINT   1 - 1    LETDD1
DRAW LETDL1 LINE    2 LETDP2   1      0
DRAW LETDL2 LINE    1 LETDP2 LETDP4
DRAW LETDL3 LINE    2 LETDP4   1      0
     LETDC2 CIRCLE  1 LETDP3   1
DRAW LETDA2 ARC     1 LETDC2   0      1571
     LETDP5 POINT   1   1      0
DRAW LETDL4 LINE    2 LETDP5   0    PAR 01

     LETL   FIGURE
     LETLP1 POINT   1 - 1    - 1
     LETLD1 DIST    7   2      1    PAR 01
DRAW LETLL1 LINE    2 LETLP1   0    LETLD1
DRAW LETLL2 LINE    2 LETLP1   2      0

     GDL    FIGURE
     GDLG   SUBFIG    LETG   - 4      0      0      1    PAR 01
     GDLD   SUBFIG    LETD     0      0      0      1    PAR 02
     GDLL   SUBFIG    LETL     4      0      0      1    PAR 03
```

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 58--63 65--70 72


     EXAMPL FIGURE
     EX1    SUBFIG    GDL       0      0      0       125 PAR 01 PAR 02 1
                      PAR 03
     EX2    SUBFIG    RECT      0      0      0       1   PAR 04
     EX3    SUBFIG    RECT      0      0      225     1   PAR 04
     EX4    SUBFIG    RECT      0      0      45      1   PAR 04
     EX5    SUBFIG    RECT      0      0      675     1   PAR 04
     EX6    SUBFIG    RECT      0      0      90      1   PAR 04
     EX7    SUBFIG    RECT      0      0      1125    1   PAR 04
     EX8    SUBFIG    RECT      0      0      135     1   PAR 04
     EX9    SUBFIG    RECT      0      0      1575    1   PAR 04


            DISPLA    EXAMPL    0      0      0       8     1      1    1
                        1       1
```

Figure 7. An Example of the Use of GDL

IV. GDL - 1

a) Introduction

As was mentioned in Section 2, an initial attempt was made using a quasi-interpretive system in which all names had been removed from the structures set up for each line in the language. The structure for each line is assigned a fixed location in memory as it is assembled, and then references to the name of that line are simply integer pointers to the location at which that line is situated.

b) Basic Structures

A/. Construction Blocks: There are six types of construction blocks of which all figures must be composed: points, line segments, circles, arcs, distances, angles. Each of these can be described in a number of ways. For example, a point can be given by two co-ordinates; this is referred to by the language as a point of kind 1. A point of kind 2 is described by the intersection of two lines. A number of consecutive words in storage is called a block. A block describing a point of kind 1 is four words long, the first three of which are used. The first indicates by the integer 1100 that a construction block of type 1 (point) kind 1 is being represented. The second word is the x co-ordinate, the third is the y co-ordinate. It is shown in this paper as in Figure 8.

A construction block for a point of kind two is given in Figure 9. Here the second word is a pointer to a block representing a line segment indicated here by L_1. This may be any kind of line segment block.

A complete list of BCU's has been given in Section III.

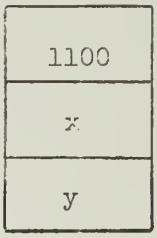


Figure 8. GDL-1 Block Structure for a Line of Kind 1

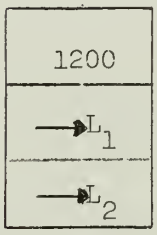


Figure 9. GDL-1 Block Structure for a Line of Kind 2

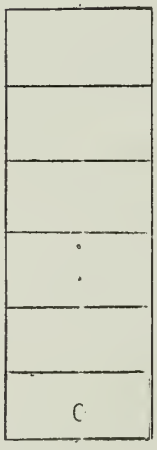


Figure 10. GDL-1 Block Structure for a Figure

B/. Figure Blocks: Those construction blocks which, when drawn, compose a figure are pointed to in a figure block. The figure block may have many pointers in it, each indicating a construction block to be drawn. The last word in a figure block is an end-of-figure indication. The paper will indicate a figure block as in Figure 10.

It should be noted that there may be many construction blocks which are not drawn; these may be considered as playing the role of construction lines. In particular, points are usually not drawn, as they would not appear significantly on a display; all distances and angles also fall into this class.

C/. Parameter and Manipulation (PAM) Blocks: A figure may have as its elements, not only construction blocks but also other figures. Each figure is initially described in terms of some origin (at the discretion of the user). When it is called as a subfigure of another figure GDL allows it to be translated by vector (x_0, y_0) rotated through θ degrees, and expanded to E times the size. Hence, the pointer in a figure calling a subfigure indicates a block giving the details of the manipulations to be done to the figure to which it in turn points. It is shown in the GDL-1 block structure as in Figure 11.

The first word in a PAM block is a pointer to the figure it is providing parameters for and manipulating. Words 2 through 5 are reserved for values of x_0, y_0, θ, and E. Following the fifth word in the PAM block are from one to twenty parameters, a word being provided for each used parameter.

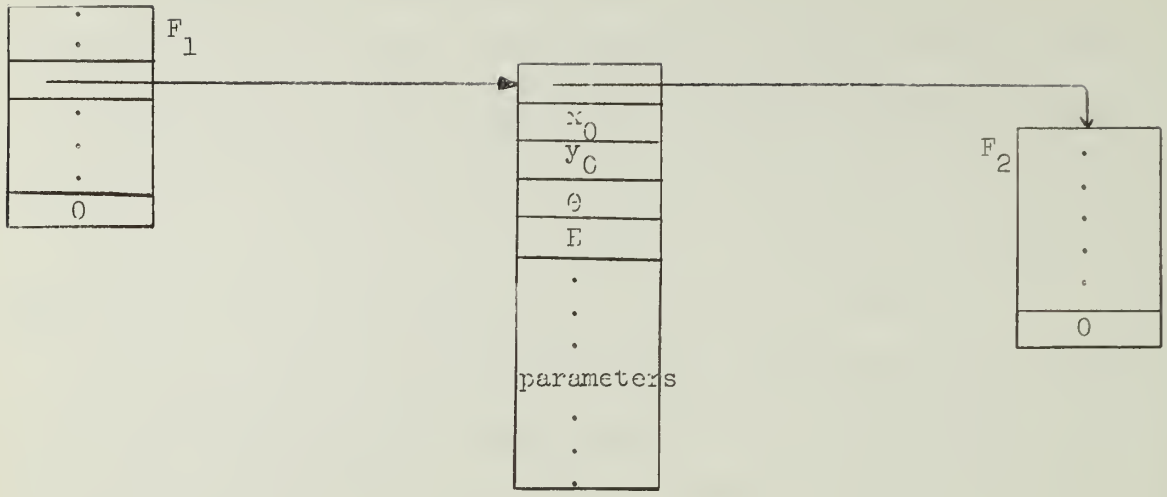


Figure 11. GDL-1 Block Structure for a PAM-Block Linkage

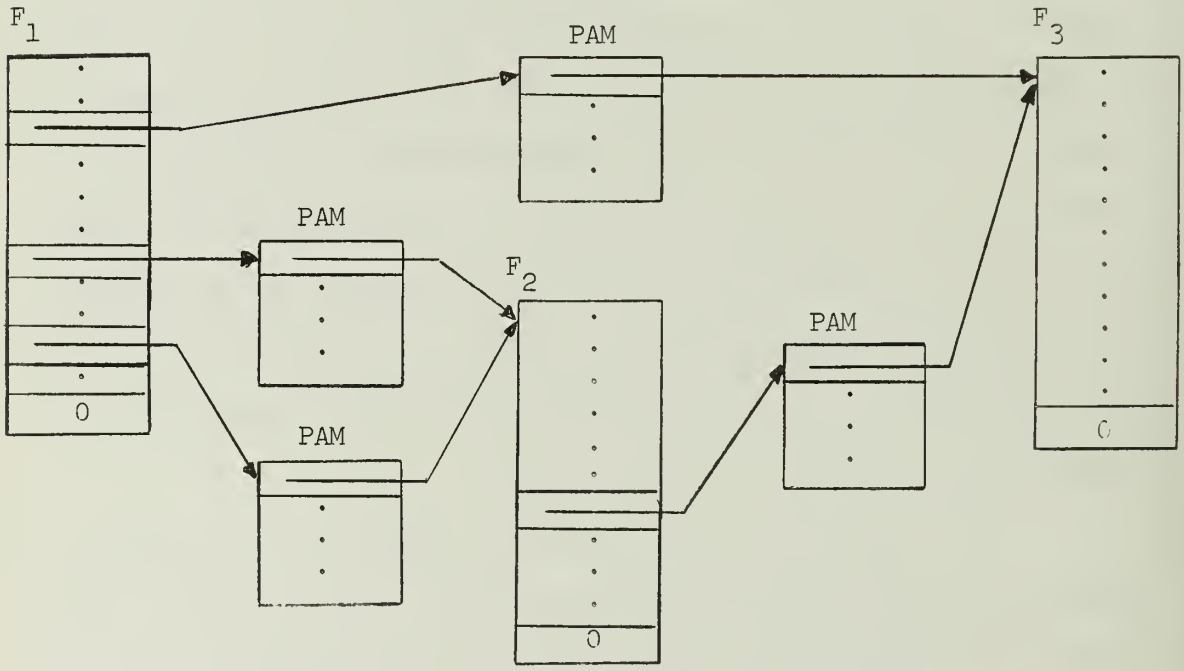


Figure 12. Nesting of Figures in GDL-1

It is often desirable to create a segment of a figure which will appear differently depending on what parameters it is fed, much like a FORTRAN subroutine. Hence, any of the construction blocks in Figure F_2 may have pointers to parameters in any words except their type-kind indicator word.

A parameter in a PAM block may be a pointer to a construction block, or it may be a value, or it may itself be a pointer to a parameter. This allows many levels of parametrization, a parameter being handed down from subfigure to subfigure through PAM blocks.

A figure may be a subfigure of a given figure many times each through a different PAM block, and it may be called as a subfigure of two different figures. Hence, patterns of GDL-1 blocks such as that shown in Figure 12 will occur. Note that F_3 will appear three times as a subfigure of F_1, twice as a part of F_2, which occurs twice in F_1.

c) Handler Commands and Effects

G.D.L. is designed to allow the user to enter the description of a drawing through an on-line CRT = display - console = typewriter - terminal, display his current description as a picture on the CRT = display, add, delete and change items in his description, redisplay, iterating until satisfied, and then obtain hard copy output from a plotter.

The language includes the following modes of operation:

A/. DEFINE: With this mode command is given the name of a new figure to be defined. A check is made to insure that the

given name has not already been used; and then it is entered in the Name Table. This table is a cross listing of names and addresses of names the user employes. No duplications are allowed.

In this mode, Basic Construction Unit Commands (BCUC) can be given. An ARC command results in a check of the items in the command for validity, and then the creation of a construction block, with cross listing in the name table. If the arc is to be drawn as part of the figure being defined, a pointer to it is added to a list of pointers for the figure.

Also a SubFigure Command (SFC) can be given in the define mode. This initiates a check for validity, then places an entry in the name table, creates a PAM - block and places a pointer to that PAM - block in the list of pointers for the figure being defined.

An END command terminates the initial definition of a figure, causes allocations of storage for the figure block, creates the figure block by copying the list of pointers into the defined space, creates the address section of the entry for the figure in the name table, and returns the program to the point of waiting for a redefinition of mode.

Validity checks resulting in the determination of errors cause the program to ignore the command, print out the cause of error, and await the next command at the same point at which the defective command was given. The error messages are given symbolically wherever possible through use of the name table.

An example is given in Figure 13.

B/. ADD: The name of the figure to which additions are to be made is given with this command. A validity check is made, and the add mode entered.

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

             .
             DEFINE
      TEST   FIGURE
             .
             END
             .
```

Figure 13. Use of DEFINE Command for Defining the Figure TEST

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

             .
      SQUARE ADD
DRAW  SQUDIA LINE     1 SQUA    SQUC
      SQUA   DUMMY
             .
             END
             .
```

Figure 14. Use of ADD Command to Add a Diagonal to the Figure SQUARE, and DUMMY to Draw Point A

In this mode BCUC and SFC commands can be used with the results indicated in the discussion of the DEFINE mode.

Also in this mode a DUMMY command can be given. This inserts a pointer to any BCU Block into the list for the figure to which additions are being made. This effectively allows the user to decide to draw a construction line.

On the END command, the pointer list generated is copied first into any blank spaces in the figure block to which the additions are being made (see discussion of the DELETE command) and then into a block of storage then allocated as a continuation of this figure block. The last word of the initial figure block is altered from the end-of-figure indicator to a pointer to the continuation block. The mode is returned to the undefined mode state.

An example is given in Figure 14.

C/. DELETE: The name of the figure from which deletions are made is given with this command; a validity check is made and the delete mode entered.

The names of Basic Construction Units and Subfigures pointed to by this figure are given, also. The pointers to these blocks in the figure block are replaced by a No-Pointer Indicator (NPI). These NPI's are replaced first if additions are made to the figure (see discussion of the ADD mode).

The mode is returned to the undefined state after each delete command.

An example is given in Figure 15.

The deletion of a BCU of SF from a figure does not remove its name from the name table, nor free the block of space used to

represent it. This is necessitated by multiple usage of BCU's. A more sophisticated storage allocation routine might justify attention to possible savings here, but at the current level it does not.

D/. CHANGE: BCUC's are given, and the name on each interpreted as being the name of the Construction Block to be altered. A convention is established to ease the change procedure: each construction block is 4 words long. Thus at the expense of a little storage it is possible to avoid the severe difficulties created by attempting to move the data for one BCUC to a different location. As many other BCU's may refer to the BCU in question, and as no information as to which these are is obtainable without exhaustive searching, the price in storage appears not unreasonable.

Thus the change mode simply does the same creation as the DEFINE and ADD modes, but places the created block where a previously defined block of the same name had been.

An END command returns the program to the undefined mode state.

An example is given in Figure 16.

E/. DISPLAY: This mode calls a subprogram which displays on a CRT the figure requested. A validity check is made and an artificial PAM - block is created to accommodate both the displaying of a figure requiring parameters and the orientation of the display.

The subprogram for creating the display is discussed under Section 4.d) Plotting, Displaying, and Representation in Fortran. Suffice it to say here that when the subprogram is completed, control

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

           .
           DELETE    SQUA    SQUDIA
           .
```

Figure 15. Use of DELETE Command. (cf. Figure 14)

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

           .
           CHANGE
     SQUB  POINT   1   0       9
     SQUC  POINT   1   8       9
           END
           .
```

Figure 16. Use of CHANGE Command to Give a Rectangle

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

           .
           DISPLA    SQUARE   0       0       0       1
           PLOT      SQUARE   0       0       0       1
           FINISH
```

Figure 17. Use of DISPLAY, PLØT and FINISH Commands

is returned to the main program leaving it in the undefined mode state.

An example is given in Figure 17.

F/. PLOT: This is identical to DISPLAY, with output on an x - y plotter rather than a CRT display.

G/. FINISH: Control is turned over to the supervisor, or a STOP command is given.

d) Plotting, Displaying, and Representation in Fortran

As the plotting and displaying were the motivation for the representation in FORTRAN as it is, and as the description of these items independent of the FORTRAN representation used would be unclear, the description of the plotting and displaying functions are to be found here together with their FORTRAN representation. In this section the term draw is used throughout. The Handler commands PLOT and DISPLAY have identical action, with the single exception that the output in the PLOT mode is generated by calling plotter routines when 'drawing' and the output in the DISPLAY mode by calling CRT display routines.

Although it is generally more economical of storage and computation time to program in assembly language, the usefulness of a package such as G.D.L. is severely limited by restricting it to one machine. Hence, it was decided to program the language in FORTRAN II using as few assembly language subroutines as possible. As even FORTRAN varies with the machine on which it is implemented, and as conversion of format internally is quite machine dependent, the

the routines which identify an input word as either a name, a value, or a parameter, and convert these to a standard representation, will generally have to be written for each machine.

To facilitate pointers, main storage is defined as a large one-dimensional array, L (1). As subscripts must be integer in FORTRAN II, the array must be integer so as to be able to hold subscripts, i.e., pointers to given locations.

On a machine with 18 bit fixed point constants, all values are stored in inches (or radians) times 1000. This allows up to about ± 30 inches (or radians) to be indicated to an accuracy of 1/1000 of an inch (or radian). In the calculating routines these values are converted to floating point for use. Here also, the range and format of variables will have to be determined for each machine.

All basic blocks are 4 words long as mentioned. A pointer appears as a positive integer. A parameter appears as a negative integer from -1 to -20, indicating which parameter it refers to. The type - kind indicator is the integer xy00 where x is the type and y is the kind.

A figure block is a series of pointers (positive integers), NPI's (the integer + 40,000), followed by the end-of-figure indicator (0). A continuation pointer is a negative integer, whose magnitude indicates the location of the continuation. As an array is limited in size to 32767 locations on the machine used, NPI's are distinguishable from pointers. A PAM - block has a negative pointer to the subfigure as its first word. This allows the drawing program to distinguish PAM - blocks from blocks representing BCU's.

If a pointer in a figure points to a location containing a positive integer the program interprets this as a type - kind indicator (xy00) which it then uses in a series of "computed - GØ - TØ" statements to locate the section of programming which calculates the values representing that type of basic construction unit. If however a pointer in a figure points to a location containing a negative integer, the program interprets it as the first word in a PAM - block. x_0, y_0, θ, E are stored as values x 1000. The parameters in a PAM - block are either pointers to blocks representing BCU's (positive integers < 30,000), pointers to parameters (negative integers from -1 to -20), or values (positive integers between 30,000 and 90,000). This last is interpreted to mean (value x 1000) + 60,000. Values in PAM - blocks are therefore limited to ±30,000 and the L(I) array limited to 29,999 locations.

The assembler program follows the outlines as suggested in Section 4. c) The Handler. A block diagram appears in Section 4. e), program listings in Section 4. f).

The drawing sub-program, employs two stacks allowing arbitrary levels of nesting of figures, or construction blocks within figures. On entering this program a pointer to the first (artificial) PAM - block is created. This is used as a base register for parameter searching and points to E. The number of the parameter desired is added to this base address to obtain the parameter. In addition from x_0, y_0, θ, E are created six values representing this manipulation. The manipulation is represented by:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S \\ T \end{pmatrix} + \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

where $\begin{pmatrix} x \\ y \end{pmatrix}$ is the vector representing the co-ordinates of a point returned by the BCU being drawn, and $\begin{pmatrix} x' \\ y' \end{pmatrix}$ is the vector of co-ordinates to be used by the displaying or plotting subroutines. Hence, from simple geometry,

$S = x_0$

$T = y_0$

$A = E \cos \theta$

$B = -E \sin \theta$

$C = E \sin \theta$

$D = E \cos \theta.$

The "next-sequential-instruction" (NSI) is set to the first pointer in the figure to be drawn (obtained from the name table). The first pointer is followed; assume a BCU is encountered. This BCU is to be drawn, but as yet the values of the BCU's upon which it depends have not been determined. Hence, a stage indicator is set to 1 and decremented as the stages are traced down. Eventually points will be reached and values obtained. Then as these values are returned up a stage, the stage indicator is augmented. When values are obtained and the stage indicator is less than 1, the values are returned up a stage through an "accumulator". When values are obtained and the stage indicator equals 1 then the results are plotted or displayed.

An example may make this clearer. Suppose it is wished to draw the line ST of Figure 18 as part of a figure named FIG1.

Points A, B, C, D, V, W are defined by co-ordinates (type-kind = 1100); lines AD, CB, VD, WB are defined as between two end-

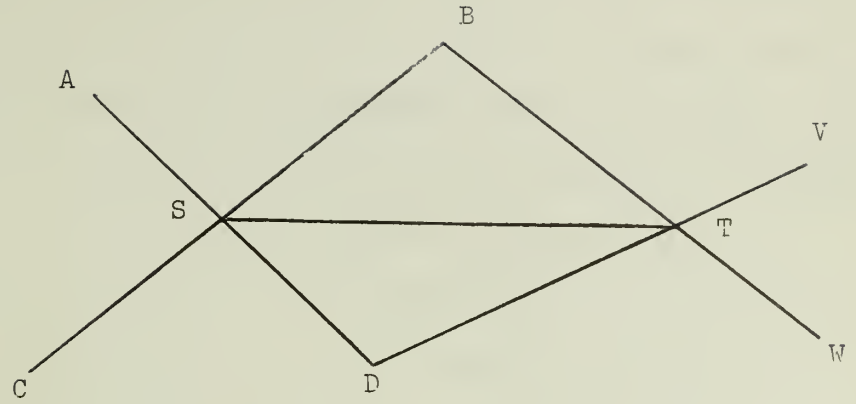


Figure 18. Diagram of Figure FIG1

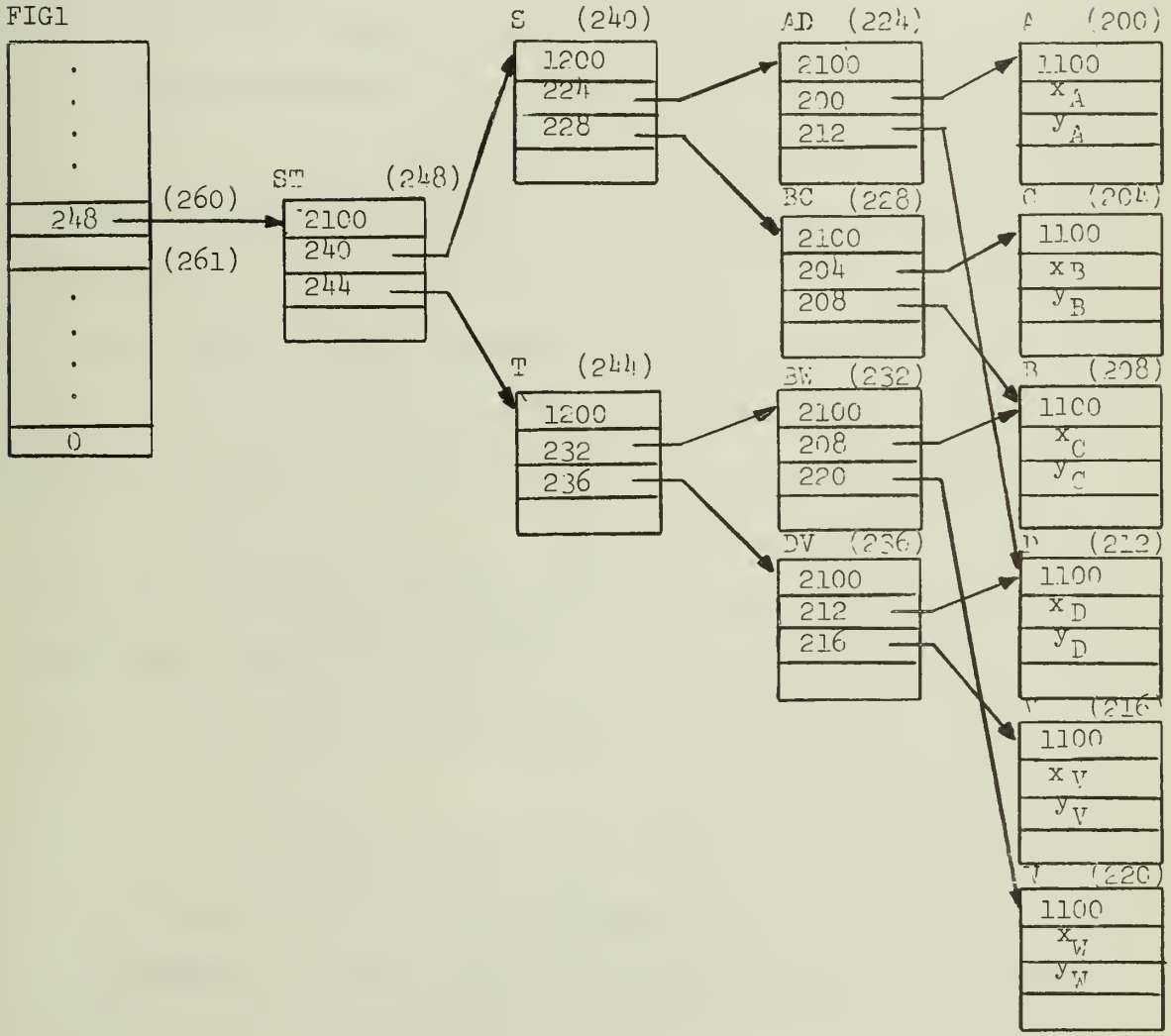


Figure 19. GDL-1 Block Structure for Figure FIG1

points (type-kind = 2100); points S and T are defined as intersections of two lines (type-kind = 1200); line ST is defined as between two end-points (type-kind = 2100). The structure resulting would be as in Figure 19. The program setting with NSI = 260 would:

a) stack 261 as NSI on return to FIG 1

b) set the stage indicator (ID) equal to 2

c) go down one stage:

i) NSI = L (NSI + LT) : pointer is followed

ii) ID = ID - 1 : stage indicator is decremented

iii) LT = 1 : program section is set for first pointer of new BCU

here; NSI = 248, ID = 1, LT = 1

d) NSI is examined, and program section 2100 is called.

e) LT = 1 indicates first pointer is to be followed. LT = 2, and NSI = 248 are stacked for return to this stage. Go down a stage, NSI = 240, ID = 0, LT = 1.

f) And so on until NSI = 200, ID = -2, OT = 1 and the stack looks like Figure 20.

g) Here, however, the program can return the values of (A_x, A_y) without searching to any deeper stages. The accumulator is loaded with these (as ID $\neq$ 1).

h) go up one stage:

i) LT = IPØP (1); pop LT from the stack

ii) NSI = IPØP (1); pop NSI from the stack

iii) ID = ID + 1 ; stage indicator is augmented

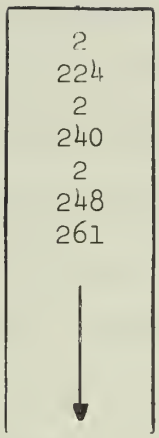


Figure 20. Stack 1 When Analyzing A of FIG1. (cf. Figure 19)

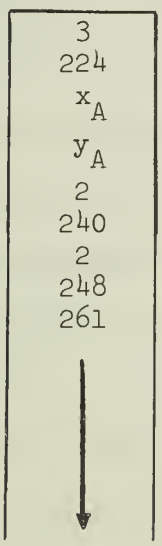


Figure 21. Stack 1 While Analyzing D of FIG1. (cf. Figure 19)

i) as NSI = 224, the segment for 2100 is again called. As LT = 2 as "computed - GØ - TØ" transfers control to that section of this segment which analyzes the second pointer.

j) again LT = 3, NSI = 224 are stacked, as are the values for point A (This last actually being done first so NSI x LT will pop first for return to the correct segment of program). The stack looks like Figure 21 and the program goes down a stage (LT = 2) : NSI = 212, LT = 1, ID = -2.

k) Again values can be immediately returned; D_x, D_y are loaded into the accumulator; the program goes up a stage: NSI = 224, LT = 3, ID = -1.

l) Now program segment 3 of program section 2100 can calculate the values for line AD. As ID = -1 it loads the accumulator, and returns up a stage.

m) The second pointer of BCU S is now followed, the values of the first being stacked as in j). A prefix scan is being carried out on the tree represented.

n) The procedure continues until NSI = 248, LT = 3, and the calculation of the values for line ST have been obtained. Now ID = 1, and so these values are passed to the plotter or display subprogram.

o) The plotter or display subprogram returns control to the drawing subprogram where it can pop NSI = 261 out of the stack and continue.

If any of the pointers in the tree described had been a parameter, instead of simply calculating NSI from L(NSI + LT), a

reference to the PAM - block calling the subfigure is made, the pointer, parameter, or value obtained from there. If it was a parameter, this procedure would have been repeated at a level higher, referring to the PAM - block calling the figure which pointed to the PAM - block in which the program found the parameter.

To clarify this and explain when subfigures are encountered while executing a figure block, the example of Figure 22 is given. FIG 1 is to be drawn. The Dummy PAM - block is created as shown. When pointer A is encountered while drawing FIG 1, the program detects that a subfigure is to be drawn. The position of E in the DUMMY PAM - block is stacked in stack 2, as indicated. The current parameter pointer is set to indicate E in PAM - block 1. Any point $\begin{pmatrix} x \\ y \end{pmatrix}$ generated by FIG 2 to be drawn as a section of FIG 1 will now be manipulated first by the x_0', y_0', θ', E', of PAM 1, and then by the x_0, y_0, θ, E of the Dummy PAM - block. But all manipulations must return to those of just the Dummy PAM - block when FIG 2 is completed. Hence the program stacks in stack 1 the current values of A, B, C, D, S, T. Then it calculates a new set of six variables to accomplish the combined manipulation. This calculation is (where the subscript "new" indicates those being calculated).

$$S_{new} = S + Ax_0' + By_0'$$

$$T_{new} = T + Cx_0' + Dy_0'$$

$$A_{new} = AE' \cos \theta' + BE' \sin \theta'$$

$$B_{new} = -AE' \sin \theta' + BE' \cos \theta'$$

$$C_{new} = CE' \cos \theta' + DE' \sin \theta'$$

$$D_{new} = -CE' \sin \theta' + DE' \cos \theta'$$

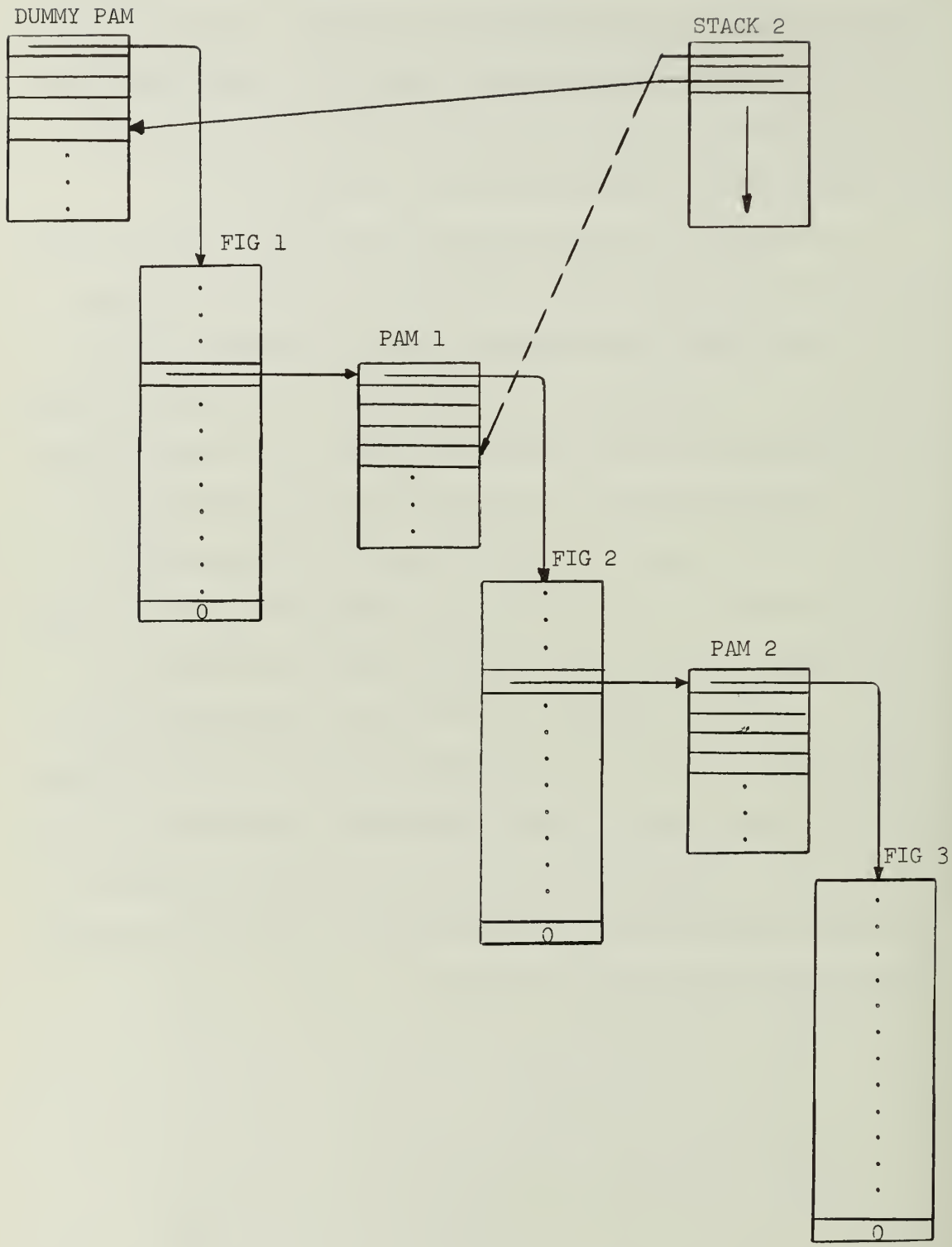


Figure 22. Nested Figures and Parameter Chasing Using Stack 2

The program then determines from the first word of PAM 1 the address of the figure to be drawn, NSI = (Location of pointer A + 1) is stacked in stack 1 and NSI is set equal to the location of the first word of FIG 2.

A similar series of actions takes place when pointer B is encountered while drawing FIG 2. When FIG 3 is being drawn, stack 2 will contain pointers to both E in PAM 1 and E in the Dummy-PAM - block; the current parameter indicator will point to the E in PAM 2.

If a BCU of the substructure of FIG 3 has a pointer to a parameter, then when it is encountered its number (say 3) is added to the current parameter indicator. If the result is a parameter (i.e. parameter 3 in PAM 2 is itself a pointer to a parameter), its number is the parameter indicator for PAM 1. This is turn may be a pointer to a parameter, in which case its value is added to the second value in stack 2, the parameter indicator for the Dummy-PAM - block. Note that a restriction is placed on parameters in the Dummy-PAM - block; they must be values. Hence no further chasing results. (Stack 2 is not strictly speaking a Push-Down Stack, the ability to read all entries in it being also provided.)

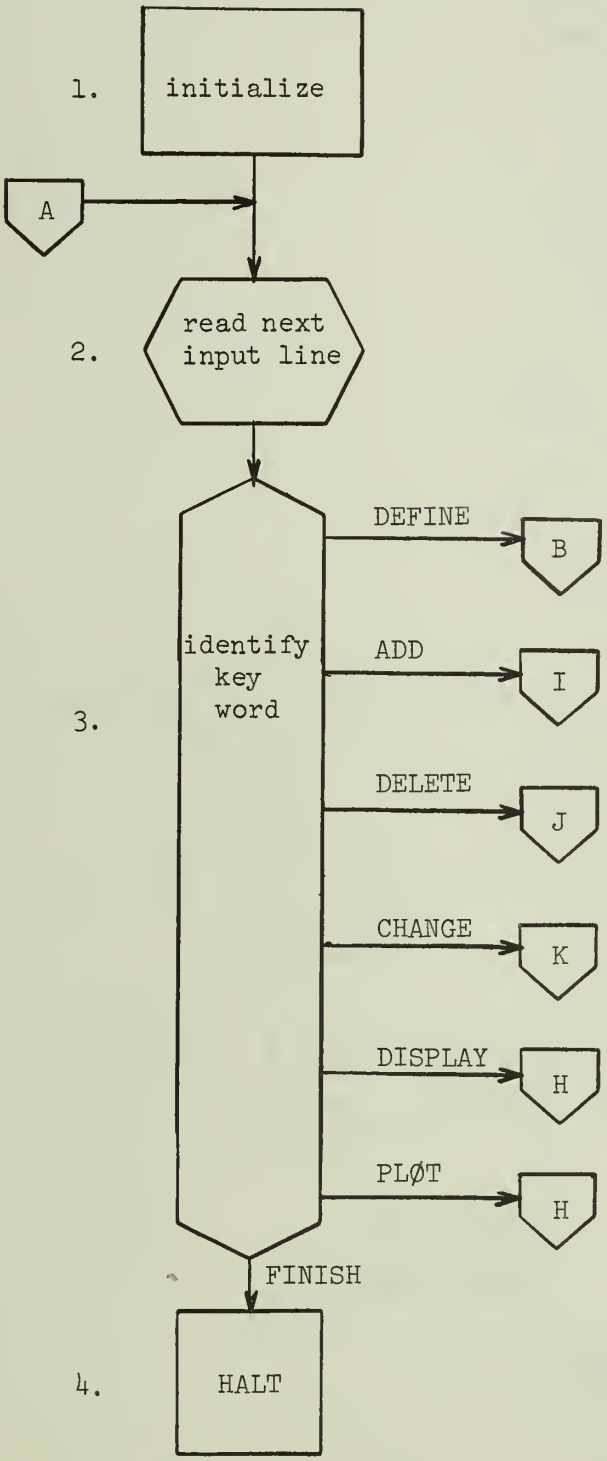
On encountering the end-of-figure indicator (a work with 0 in it) of FIG 3, the current parameter indicator is set equal to the popped top entry in stack 2. NSI is set equal to the popped entry from stack 1, A, B, C, D, S, T are replaced by the six old values popped from stack 1, and processing continues.

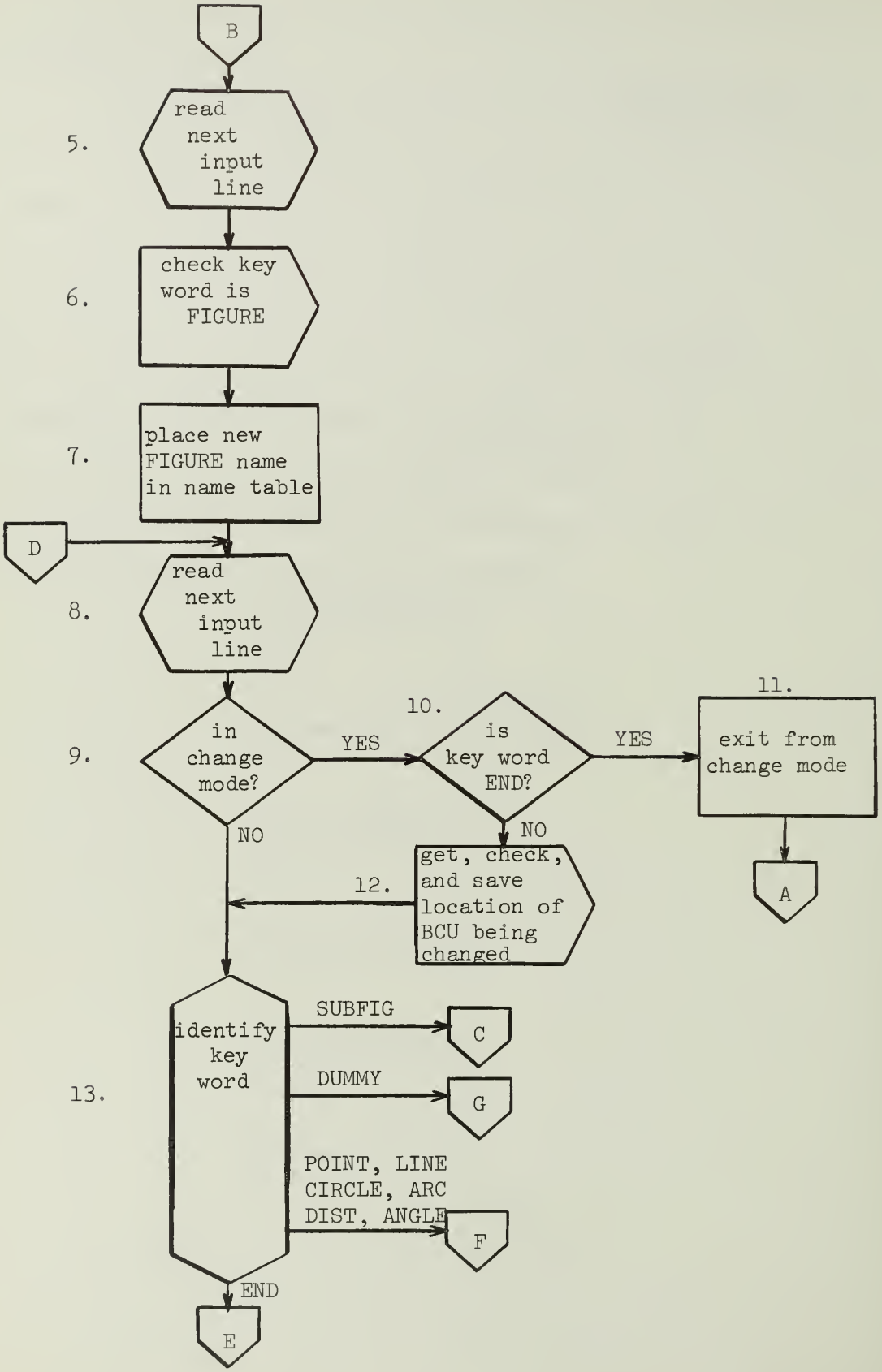
When the value of the entry popped from stack 2 is the known value of the location of E in the Dummy-PAM - block (this is fixed at

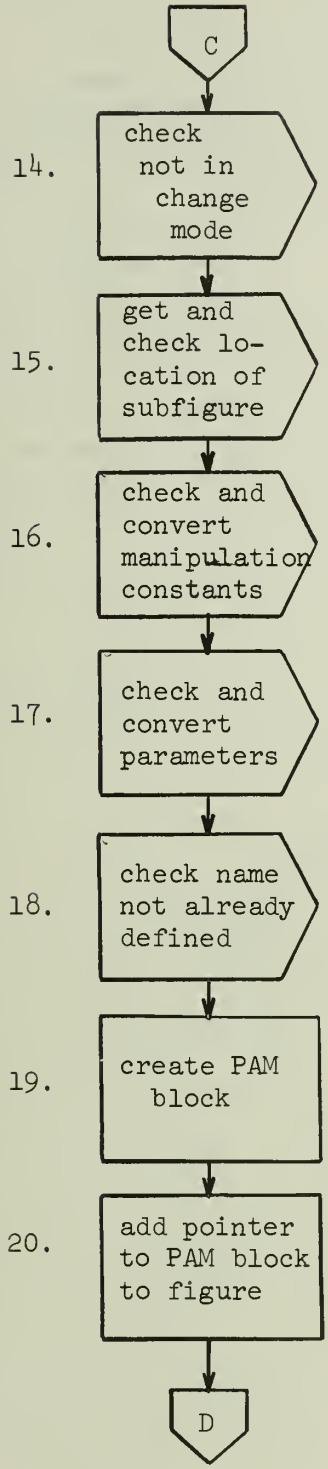
0) the drawing sub-program returns control to the G.D.L. assembler at the undefined mode stage.

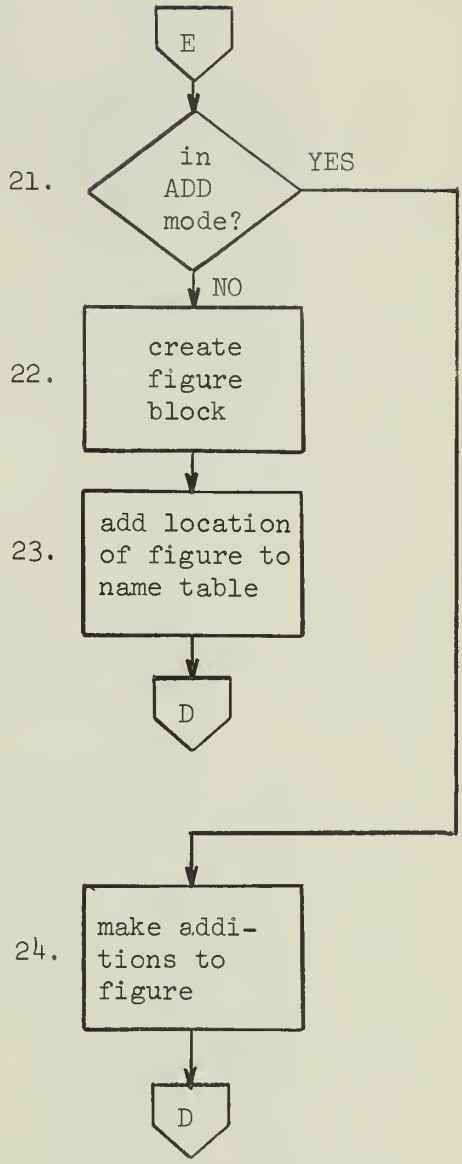
Storage Allocation has been left in a most crude state. All blocks are assigned sequentially, and there is no recourse of action if core is exceeded. As arrays are in FORTRAN, and hence lack dynamic allocation facilities, maximum stack sizes must be guessed at, and space wasted while it is not fully in use. If it is exceeded, again there is no way out. This seems to be the most serious price one pays for working in FORTRAN. Minimum block size is four words. All BCU's have four words. If a one-entry addition is made to a figure block, an additional block of four words is provided for chaining. This prevents many small blocks going to waste.

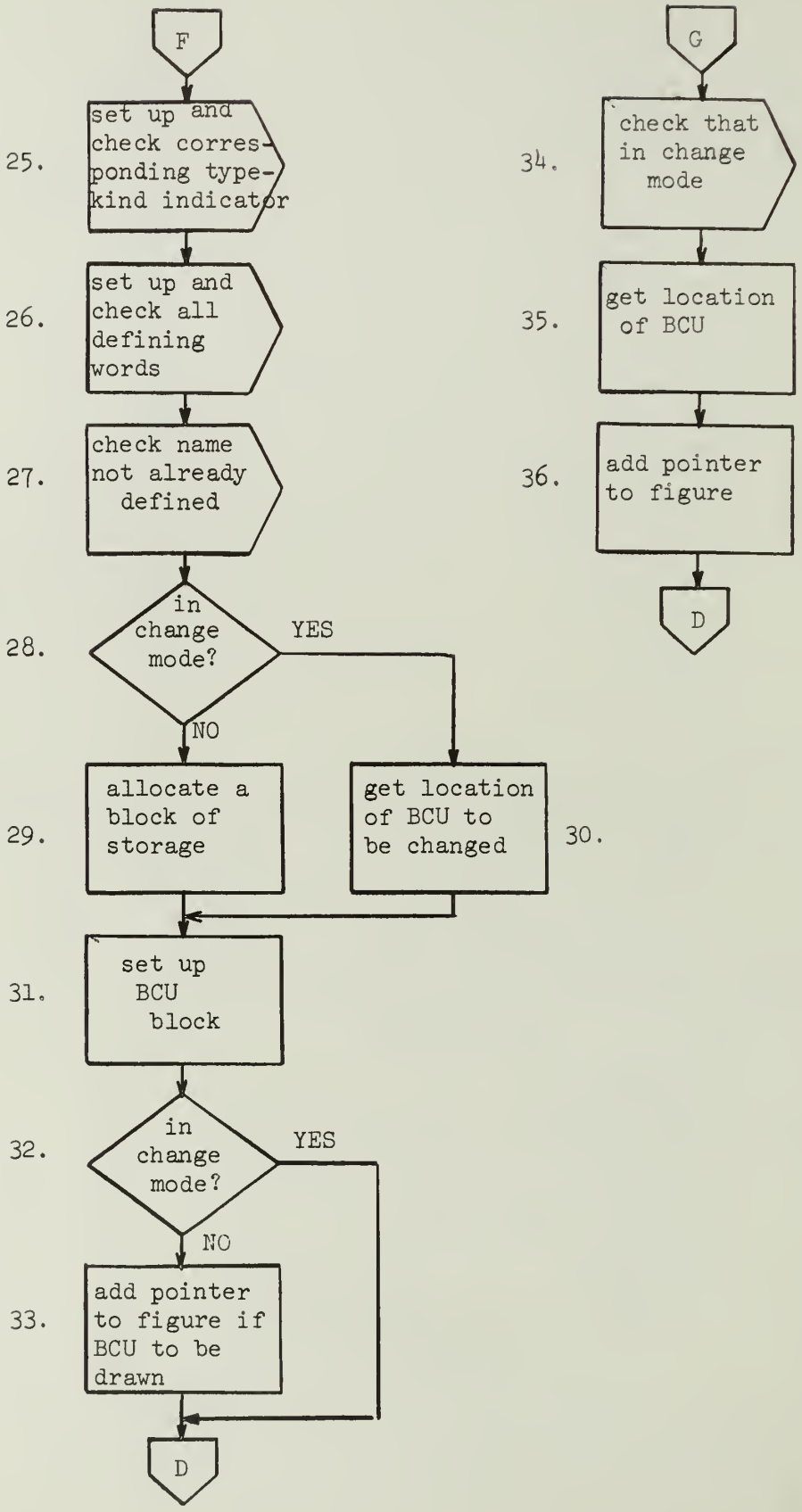
e) Block Diagrams

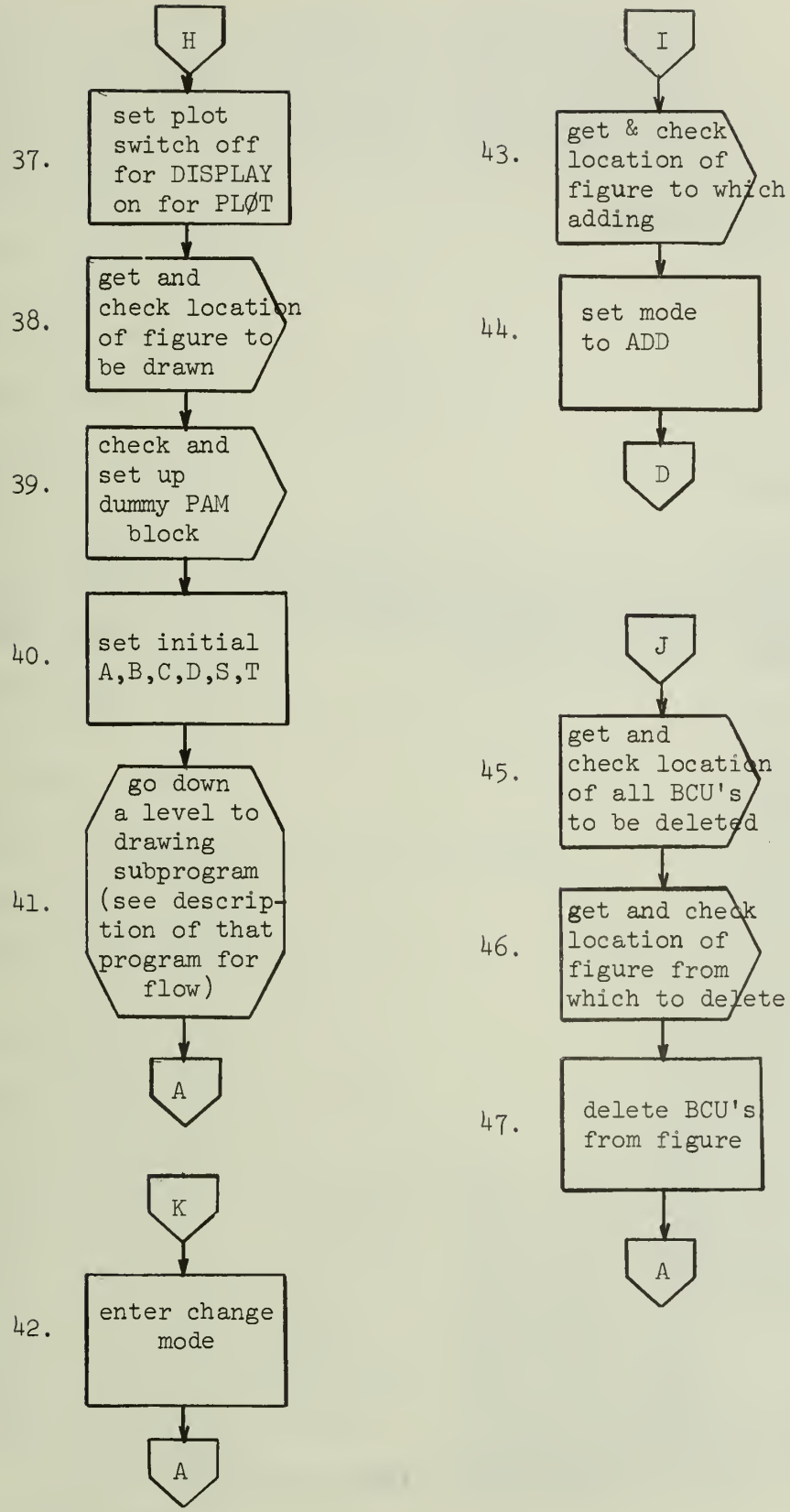



B
5.
read next input line
6.
check key word is FIGURE
7.
place new FIGURE name in name table
D
8.
read next input line
9.
in change mode?
YES
10.
is key word END?
YES
11.
exit from change mode
A
NO
NO
12.
get, check, and save location of BCU being changed
13.
identify key word
SUBFIG
C
DUMMY
G
POINT, LINE
CIRCLE, ARC
DIST, ANGLE
F
END
E


C
14.
check not in change mode
15.
get and check location of subfigure
16.
check and convert manipulation constants
17.
check and convert parameters
18.
check name not already defined
19.
create PAM block
20.
add pointer to PAM block to figure
D


E
21.
in ADD mode?
YES
NO
22.
create figure block
23.
add location of figure to name table
D
24.
make additions to figure
D


F
25.
set up and check corresponding type-kind indicator
26.
set up and check all defining words
27.
check name not already defined
28.
in change mode?
YES
NO
29.
allocate a block of storage
30.
get location of BCU to be changed
31.
set up BCU block
32.
in change mode?
YES
NO
33.
add pointer to figure if BCU to be drawn
D
G
34.
check that in change mode
35.
get location of BCU
36.
add pointer to figure
D


H
37.
set plot switch off for DISPLAY on for PLØT
38.
get and check location of figure to be drawn
39.
check and set up dummy PAM block
40.
set initial A,B,C,D,S,T
41.
go down a level to drawing subprogram (see description of that program for flow)
A
K
42.
enter change mode
A
I
43.
get & check location of figure to which adding
44.
set mode to ADD
D
J
45.
get and check location of all BCU's to be deleted
46.
get and check location of figure from which to delete
47.
delete BCU's from figure
A

f) Program Listing

```
C      PROGRAM  GDL-1
C
C
       DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
       DIMENSION NTL(100,2),LBLK(80)
       COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
       COMMON A,B,C,D,S,T,LIN
C
C   INITIALIZE
C
       LPT1=0                                                    1
       LPT2=0
       LALLPT=25
       LADD=0
       LCHANG=0
       NTLPT=0
       DO 3601 I=1,80
 3601  LBLK(I)=3
       LBLK(11)=2
       LBLK(12)=2
       LBLK(14)=1
       LBLK(21)=2
       LBLK(61)=2
       LBLK(63)=2
       LBLK(64)=2
       LBLK(65)=1
       LBLK(66)=1
       LBLK(71)=2
       LBLK(73)=1
C
C G.D.L.-1 HANDLER
C
 3000  CALL RDLN(I)                                              2
       IF(I) 3003,3003,9002
 3003  IF( LIN(2,1)-6HDEFINE) 3150,3010,3150                     3
```

```
 3150 IF( LIN(2,1)-6HADD    ) 3151,3300,3151
 3151 IF( LIN(2,1)-6HDELETE) 3152,3400,3152
 3152 IF( LIN(2,1)-6HCHANGE) 3001,3500,3001
 3001 IF( LIN(2,1)-6HDISPLA) 3002,3100,3002
 3002 IF( LIN(2,1)-6HPLOT   ) 3252,3250,3252
 3252 IF( LIN(2,1)-6HFINISH) 3009,3200,3009
 3009 PRINT  10001
      GO TO 3000
 9002 PRINT 10002
      GO TO 3000
C
C FINISH ORDER
C
 3200 STOP 02                                                    4
C
C DEFINE COMMAND
C
 3010 CALL RDLN(I)                                               5
      IF(I) 3004,3004,9003
 3004 LADD=0
 3012 IF( LIN(2,1)-6HFIGURE) 3013,3014,3013                      6
 3013 PRINT 10004
      GO TO 3010
 9003 PRINT 10002
      GO TO 3010
C
C FIGURE COMMAND
C
 3014 LT=LIN(1,2)                                                7
      IF(LSRCHN(LT)) 9005,3026,9005
 9005 PRINT 10005
      GO TO 3010
 3026 NTLPT=NTLPT+1
      NTL(NTLPT,1)=LT
      IX=0
 3015 CALL RDLN(I)                                               8
      IF(I) 3005,3005,9006
 9006 PRINT 10002
      GO TO 3015
 3005 IF(LCHANG) 3025,3025,3504                                  9
 3025 IF( LIN(2,1)-6HSUBFIG) 3016,3030,3016                     13
 3016 IF( LIN(2,1)-6HDUMMY ) 3024,3600,3024
 3024 IF( LIN(2,1)-6HPOINT ) 3017,3060,3017
 3017 IF( LIN(2,1)-6HLINE  ) 3018,3080,3018
 3018 IF( LIN(2,1)-6HCIRCLE) 3019,3082,3019
 3019 IF( LIN(2,1)-6HARC   ) 3020,3084,3020
 3020 IF( LIN(2,1)-6HDIST  ) 3021,3090,3021
 3021 IF( LIN(2,1)-6HANGLE ) 3022,3086,3022
 3022 IF( LIN(2,1)-6HEND   ) 3023,3050,3023
 3023 PRINT 10006
      GO TO 3015
C
C SUBFIG ORDER
```

```
C
 3030 IF(LCHANG) 3520,3520,9008                                    14
 9008 PRINT 10008
      GO TO 3015
 3520 IF(LIN(3,2)) 3031,9009,9009
 9009 PRINT 10009
      GO TO 3015
 3031 LST1(1)=-LSRCHN(LIN(3,1))                                    15
      IF(LST1(1)) 9010,9010,3032
 9010 PRINT 10010
      GO TO 3015
 3032 IF(LIN(4,2)) 9011,3033,9011                                  16
 3033 IF(LIN(5,2)) 9011,3034,9011
 3034 IF(LIN(6,2)) 9011,3035,9011
 3035 TEMP=LIN(6,1)
      TEMP=TEMP/57.29578
      LIN(6,1)=TEMP
      IF(LIN(7,2)) 9011,3036,9011
 9011 PRINT 10011
      GO TO 3015
 3036 K=LIN(2,2)+2
      IF(K-7) 3039,3039,3040
 3040 DO 3037 I=8,K                                                17
      IF(LIN(I,2)) 3038,3780,3781
 3038 LST1(I)= LSRCHN(LIN(I,1))
      IF(LST1(I)) 9012,9012,3037
 9012 PRINT 10012
      GO TO 3015
 3780 LST1(I)=LIN(I,1)+60000
      GO TO 3037
 3781 LST1(I)=-LIN(I,1)
 3037 CONTINUE
 3039 IF(LSRCHN(LIN(1,2))) 9013,3046,9013                          18
 9013 PRINT 10013
      GO TO 3015
 3046 N=LALLOC(K-2)                                                19
      NTLPT=NTLPT+1
      NTL(NTLPT,1)=LIN(1,2)
      NTL(NTLPT,2)=N
      L(N)=-LST1(1)
      L(N+1)=LIN(4,1)
      L(N+2)=LIN(5,1)
      L(N+3)=LIN(6,1)
      L(N+4)=LIN(7,1)
      IF(K-7) 3047,3047,3041
 3041 DO 3042 I=8,K
      I1=N+I-3
 3042 L(I1)=LST1(I)                                                20
 3047 IX=IX+1
      LST2(IX)=N
      GO TO 3015
C
C END ORDER
```

```
C
 3050 IF(IX) 9014,9014,3051
 9014 PRINT 10014
      GO TO 3015
 3051 IF(LADD) 3053,3053,3056                                    21
 3053 K=IX+1                                                     22
      N=LALLOC(K)
      DO 3052 I=1,IX
      K=N+I-1
 3052 L(K)=LST2(I)
      K=N+IX
      L(K)=0
      DO 3054 K=1,NTLPT                                          23
      IF(NTL(K,1)-LT) 3054,3055,3054
 3054 CONTINUE
 3055 NTL(K,2)=-N
      GO TO 3000
 3056 I=1                                                        24
 3310 M=LT
 3313 IF(L(LT)) 3057,3059,3058
 3057 LT=-L(LT)
      GO TO 3310
 3058 IF(L(LT)-40000) 3311,3312,3311
 3311 LT=LT+1
      GO TO 3313
 3312 L(LT)=LST2(I)
      I=I+1
      IF(IX-I) 3314,3311,3311
 3314 LADD=0
      GO TO 3000
 3059 K=3-LT+M
      IF(K) 3316,3316,3317
 3317 L(LT+1)=0
      GO TO 3312
 3316 K=IX-I+2
      N=LALLOC(K)
      L(LT)=-N
      DO 3315 J=I,IX
      K=N+J-I
 3315 L(K)=LST2(J)
      K=N+IX-I+1
      L(K)=0
      GO TO 3314
C
C POINT ORDER
C
 3060 M=LIN(2,2)+10                                              25
 3063 K=LIN(7,1)
      IF(LBLK(M)-K) 9015,3064,9015
 9015 PRINT 10015
      GO TO 3015
 3064 DO 3069 I=1,K                                              26
      IF(LIN(I+2,2)) 3065,3066,3067
```

```
 3065 N=LSRCHN(LIN(I+2,1))
      IF(N) 9016,9016,3068
 9016 PRINT 10016
      GO TO 3015
 306  LST1(I)=N
      GO TO 3069
 3066 LST1(I)=LIN(I+2,1)+60000
      GO TO 3069
 3067 LST1(I)=-LIN(I+2,1)
 3069 CONTINUE
      IF(LSRCHN(LIN(1,2))) 9017,3070,9017                    27
 9017 PRINT 10017
      GO TO 3015
 3070 IF(LCHANG) 3508,3508,3507                              28
 3508 N=LALLOC(4)                                            29
      NTLPT=NTLPT+1
      NTL(NTLPT,1)=LIN(1,2)
      NTL(NTLPT,2)=N
 3509 L(N)=M*100                                             31
      K=LIN(7,1)
      DO 3768 I=1,K
      M=N+I
 3768 L(M)=LST1(I)
      IF(LCHANG) 3510,3510,3015                              32
 3510 IF( LIN(1,1)-6HDRAW  ) 3015,3769,3015                  33
 3769 IX=IX+1
      LST2(IX)=N
      GO TO 3015
C
C  LINE ORDER
C
 3080 M=LIN(2,2)+20                                          25
      GO TO 3063
C
C  CIRCLE ORDER
C
 3082 M=LIN(2,2)+40                                          25
      GO TO 3063
C
C  ARC ORDER
C
 3084 M=LIN(2,2)+50                                          25
      GO TO 3063
C
C  DISTANCE ORDER
C
 3090 M=LIN(2,2)+60                                          25
      GO TO 3063
C
C  ANGLE ORDER
C
 3086 M=LIN(2,2)+70                                          25
      GO TO 3063
```

```
C
C DUMMY ROUTINE
C
 3600 IF(LCHANG) 9018,9018,3503                                    34
 9018 PRINT 10018
      GO TO 3015
 3530 N=LSRCHN(LIN(1,2))                                           35
      IF(N) 9020,9020,3769
 9020 PRINT 10020
      GO TO 3015                                                   36
C
C DISPLAY ORDER
C
 3100 LPLTSW=0                                                     37
 3251 IF(LIN(3,2)) 3101,3009,3009
 3101 NSI=-LSRCHN(LIN(3,1))                                        38
      IF(NSI) 3009,3009,3102
 3102 K=LIN(2,2)                                                   39
      IF(K-5) 3009,3105,3106
 3106 DO 3104 I=6,K
      IF(LIN(I+2,2)) 3009,3103,3009
 3103 L(I-5)=LIN(I+2,1)
 3104 CONTINUE
 3105 DO 3108 I=1,4
      IF(LIN(I+3,2)) 3009,3108,3009
 3108 L(I+20)=LIN(I+3,1)
      N=20                                                         40
      CALL STACK2(0)
      S=0
      T=0
      A=1
      B=0
      C=0
      D=1
      GO TO 0031                                                   41
C
C PLOT ORDER
C
 3250 LPLTSW=1                                                     37
      GO TO 3251
C
C ADD ORDER
C
 3300 LT=-LSRCHN(LIN(1,2))                                         43
      IF(LT)9023,9023,3302
 9023 PRINT 10023
 9023 GO TO 3000
 3302 IX=0                                                         44
      LADD=1
      GO TO 3015
C
C DELETE ORDER
C
```

```
 3400 LT=LIN(2,2)                                                    45
      DO 3401 I=1,LT
      IF(LIN(I+2,2)) 3402,9024,9024
 9024 PRINT 10024
      GO TO 3000
 3402 LST2(I)=LSRCHN(LIN(I+2,1))
      IF(LST2(I)) 3401,9025,3401
 9025 PRINT 10025
      GO TO 3000
 3401 CONTINUE
      K=LSRCHN(LIN(1,2))                                             46
      IF(K) 3408,9026,9026
 9026 PRINT 10026
      GO TO 3000
 3408 DO 3403 I=1,LT                                                 47
      M=LST2(I)
      N=-K
 3406 IF(L(N)) 3404,3403,3405
 3404 N=-L(N)
      GO TO 3406
 3405 IF(L(N)-M) 3409,3407,3409
 3409 N=N+1
      GO TO 3406
 3407 L(N)=40000
 3403 CONTINUE
      GO TO 3000
C
C CHANGE ORDER
C
 3500 LCHANG =1                                                      42
      GO TO 3015
 3504 IF(LIN(2,1)-6HEND    ) 3260,3503,3260                          10
 3503 LCHANG=0                                                       11
      GO TO 3000
 3260 N=LSRCHN(LIN(1,2))                                             12
      IF(N) 9021,9021,3505
 9021 PRINT 10021
      GO TO 3015
 3505 IF(L(N)) 9022,3506,3506
 9022 PRINT 10022
      GO TO 3015
 3506 LCH=N
      LIN(1,2)=0
      GO TO 3024
 3507 N=LCH                                                          30
      GO TO 3509
C
C     HANDLER ERROR MESSAGES
C
10001 FORMAT(1H ,40HILLEGAL KEY WORD. DOES NOT DEFINE MODE  )
10002 FORMAT(1H ,20HINVALID INPUT LINE  )
10004 FORMAT(1H ,40HFIGURE COMMAND DOES NOT FOLLOW  DEFINE  )
10005 FORMAT(1H ,40HNAME IN FIGURE COMMAND NOT A NAME       )
```

```
10006 FORMAT(1H ,40HILLEGAL KEY WORD. DOES NOT DEFINE A LINE)
10008 FORMAT(1H ,40HSUBFIG COMMAND ILLEGAL IN CHANGE MODE    )
10009 FORMAT(1H ,40HSUBFIGURE NAME WORD NOT A NAME           )
10010 FORMAT(1H ,40HSUBFIGURE NAME NOT IN NAME TABLE         )
10011 FORMAT(1H ,40HNCN-NUMERIC MANIPULATION CONSTANT IN SUB
     1           ,20HFIG COMMAND          )
10012 FORMAT(1H ,40HNAME IN PARAMETER LIST INDEFINED         )
10013 FORMAT(1H ,40HNAME OF SUBFIGURE ALREADY DEFINED        )
10014 FORMAT(1H ,40HNC LINES DEFINED IN THIS FIGURE          )
10015 FORMAT(1H ,40HINVALID KIND INDICATOR WORD              )
10016 FORMAT(1H ,40HPOINTER NAME NOT IN NAME TABLE           )
10017 FORMAT(1H ,40HB.C.U. NAME ALREADY DEFINED              )
10018 FORMAT(1H ,40HDUMMY COMMAND ILLEGAL IN CHANGE MODE     )
10020 FORMAT(1H ,40HNAME IN DUMMY COMMAND NOT DEFINED        )
10021 FORMAT(1H ,40HNAME IN B.C.U. NOT DEFINED IN CHANGE MOD
     1           , 1HE)
10022 FORMAT(1H ,40HNAME DOES NOT POINT TO A B.C.U.          )
10023 FORMAT(1H ,40HFIGURE NAME NOT DEFINED WHILE ADDING     )
10024 FORMAT(1H ,40HBCU NAME TO BE DELETED NOT A NAME        )
10025 FORMAT(1H ,40HBCU NAME TO BE DELETED NOT DEFINED       )
10026 FORMAT(1H ,40HNAME TO DELETE FROM NOT DEFINED OR A BCU)
C
C
C
C       DRAWING SUBPROGRAM SEGMENTS
C
C
C  EXECUTE FIGURE BLOCK
C
 0011 NSI=IPOP(1)
 0012 IF(L(NSI)) 0013,0020,0014
 0013 NSI=-L(NSI)
      GO TO 0012
 0014 IF(L(NSI)-40000) 0015,0016,0015
 0016 NSI=NSI+1
      GO TO 0012
 0015 N=L(NSI)
      IF(L(N)) 0030,0017,0017
 0017 CALL STACK1(NSI+1)
      ID=1
      LT=1
      NSI=N
      GO TO 1
C
C      END OF FIGURE BLOCK
C
 0020 LT=IPOP(2)
      IF(LT)0021,3000,0021
 0021 NSI=IPOP(1)
      T=IPOP(1)
      T=T/1000.
      S=IPOP(1)
      S=S/1000.
```

```
      A=IPOP(1)
      A=A/1000.
      B=IPOP(1)
      B=B/1000.
      C=IPOP(1)
      C=C/1000.
      D=IPOP(1)
      D=D/1000.
      GOTO 0012
C
C     GO DOWN FIG LEVEL
C
 0030 LT=D*1000.
      CALL STACK1(LT)
      LT=C*1000.
      CALL STACK1(LT)
      LT=B*1000.
      CALL STACK1(LT)
      LT=A*1000.
      CALL STACK1(LT)
      LT=S*1000.
      CALL STACK1(LT)
      LT=T*1000.
      CALL STACK1(LT)
      CALL STACK1(NSI+1)
      CALL STACK2(N+4)
      NSI=-L(N)
 0031 TH=L(N+3)
      TH=TH/1000.
      X1=COS(TH)
      X2=SIN(TH)
      X3=L(N+4)
      X3=X3/1000.
      Y1=L(N+1)
      Y1=Y1/1000.
      Y2=L(N+2)
      Y2=Y2/1000.
      S=S+A*Y1+B*Y2
      T=T+C*Y1+D*Y2
      Y3=X3*(A*X1+B*X2)
      B=X3*(-A*X2+B*X1)
      A=Y3
      Y3=X3*(C*X1+D*X2)
      D=X3*(-C*X2+D*X1)
      C=Y3
      GOTO  0012
C
C     ENTERING A LOWER LEVEL ROUTINE
C
 0002 CALL ICHASE(NSI+LT,N,NPAR)
      IF(NPAR) 0099,0099,0004
 0005 NPAR=-NPAR
 0004 CALL STACK1(LT)
```

```
      CALL STACK1(NSI)
      ID = ID-1
      LT = 1
      NSI = NPAR
      GOTO 0001
C
C     ENTERING HIGHER LEVEL ROUTINE
C
 0003 ID = ID+1
      NSI=IPOP(1)
      LT=IPOP(1)
      LT=LT+1
      GO TO 1
C
C            CALCULATING ROUTINES
C
C
C               CALCULATING ROUTINE ERROR ROUTINE
C
 0099 PRINT 13001
13001 FORMAT(1H ,17HERROR IN ROUTINES)
      CALL LSRCHP(NSI,NPAR)
      PRINT 96,LT,NPAR
   96 FORMAT(1H ,14HVALUE IN WORD ,I2,4H OF ,A6)
      GO TO 3000
C
C     CHOOSE ROUTINE FOR SPECIFIC FIGURE
C
 0001 KSN = L(NSI)
      NUM = KSN/1000
      KSN = (KSN-NUM*1000)/100
      GO TO  (1000,2000,9000,4000,5000,6000,7000),NUM
 9000 LT=0
      GO TO 99
C
C     CHOOSE ROUTINE FOR SPECIFIC POINT
C
 1000 GOTO (1100,1200,1300,1400),KSN
C
C
C     ROUTINE FOR POINT DEFINED BY TWO COORDINATES
C
 1100 GO TO (1110,1120,1130),LT
 1110 CALL IVAL(NSI+1,NPAR)
      IF(NPAR) 0005,0099,1111
 1111 LT=2
      MACC1=NPAR-60000
 1120 CALL STACK1(MACC1)
      CALL IVAL(NSI+2,NPAR)
      IF(NPAR) 0005,0099,1121
 1121 MACC1=NPAR-60000
 1130 MACC2=MACC1
      LT=LT+1
```

```
      MACC1=IPOP(1)
      GO TO 1050
C
C
C     ROUTINE FOR POINT DEFINED BY INTERSECTION OF TWO LINES
C
 1200 GO TO (1210,1220,1230),LT
 1210 N=1
      GO TO 2
 1220 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 1230 X2=MACC1
      Y2=MACC2
      X1=IPOP(1)
      Y1=IPOP(1)
      DX1=X2-X1
      DX2=X4-X3
      IF(DX1) 1238,1234,1238
 1238 IF(DX2) 1231,1236,1231
 1231 XM1=(Y2-Y1)/DX1
      XM2=(Y4-Y3)/DX2
      DM=XM1-XM2
      IF(ABSF(DM)-1.) 1232,99,1232
 1232 MACC2=(Y3-XM2*X3+XM1*X1-Y1)/DM +0.5
      MACC1=(MACC2-Y3+XM2*X3)/XM2 +0.5
 1237 IF(ABSF(MACC2)-30000) 1233,99,99
 1233 IF(ABSF(MACC1)-30000)1050,99,99
 1234 IF(DX2) 1235,99,1235
 1235 MACC1=X1 +0.5
      MACC2=Y3+(Y4-Y3)/DX2*(X1-X3) +0.5
      GO TO 1237
 1236 MACC1=X3
      MACC2=Y1+(Y2-Y1)/DX1*(X3-X1) +0.5
      GO TO 1237
C
C
C     ROUTINE FOR POINT DEFINED BY TRANSLATION FROM A POINT
C
 1300 GOTO(1310,1320,1330,1340),LT
 1310 N=1
      GO TO 0002
 1320 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      CALL IVAL(NSI+LT,NPAR)
      IF(NPAR) 5,99,1321
 1321 MACC1=-NPAR-60000
      LT=LT+1
 1330 CALL STACK1(MACC1)
      CALL IVAL(NSI+LT,NPAR)
      IF(NPAR) 5,99,1331
 1331 MACC1=-NPAR-60000
```

```
      LT=LT+1
 1340 N=IPOP(1)+IPOP(1)
      MACC2=MACC1+IPOP(1)
      MACC1=N
      GO TO 1050
C
C
C     ROUTINE FOR PT DEFINED BY CIRCLE
C
 1400 GO TO (1410,1420),LT
 1410 N=4
      GO TO 2
 1420 GO TO 1050
C
C
C      ROUTINE TO DRAW A POINT
C
 1050 IF(ID-1)  3,1051, 3
 1051 IF(LPLTSW) 1052,1052,1053
 1052 CONTINUE
C    ROUTINE TO DISPLAY A POINT
      GO TO 11
 1053 CONTINUE
C   ROUTINE TO PLOT A POINT
      GO TO 11
C
C
C     CHOOSE ROUTINE FOR SPECIFIC LINE
C
 2000 GOTO (2100,2200),KSN
C
C
C     ROUTINE FOR LINE SEGMENT DEFINED BY TWO POINTS
C
 2100 GO TO (2110,2120,2130),LT
 2110 N=1
      GO TO 2
 2120 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 2130 MACC3=MACC1
      MACC4=MACC2
      MACC1=IPOP(1)
      MACC2=IPOP(1)
      GO TO 2050
C
C
C          LINE SEGMENT DEFINED BY POINT AND TRANSLATION
C
 2200 GO TO (2210,2220,2230,2240),LT
 2210 N=1
      GO TO 2
```

```
 2220 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      CALL IVAL(NSI+LT,NPAR)
      IF(NPAR) 5,99,2221
 2221 MACC1=NPAR-60000
      LT=LT+1
 2230 CALL STACK1(MACC1)
      CALL IVAL(NSI+LT,NPAR)
      IF(NPAR) 5,99,2231
 2231 MACC1=NPAR-60000
      LT=LT+1
 2240 N=IPOP(1)+IPOP(1)
      MACC2=MACC1+IPOP(1)
      MACC1=MACC2
      GO TO 2050
C
C
C       DRAW A LINE SEGMENT
C
 2050 IF(ID-1)  3,2051, 3
 2051 IF(LPLTSW) 2052,2052,2053
 2052 CONTINUE
C     ROUTINE TO DISPLAY A LINE SEGMENT
      GO TO 11
 2053 CONTINUE
C     ROUTINE TO PLOT A LINE SEGMENT
      GO TO 11
C
C
C     CHOOSE ROUTINE FOR SPECIFIC CIRCLE
C
 4000 GOTO (4100,4200,4300),KSN
C
C
C     ROUTINE FOR CIRCLE DEFINED BY PT AND DIST
C
 4100 GO TO (4110,4120,4130),LT
 4110 N=1
      GO TO 2
 4120 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      CALL IDIST(NSI+LT,NPAR)
      IF(NPAR) 4121,99,4
 4121 MACC1=-NPAR-60000
      LT=LT+1
 4130 MACC3=MACC1
      MACC1=IPOP(1)
      MACC2=IPOP(1)
      GO TO 4050
C
C
C     ROUTINE FOR CIRCLE DEFINED BY CENTER AND PT ON CIRCUM
C
```

```
 4200 GO TO (4210,4220,4230),LT
 4210 N=1
      GO TO 2
 4220 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 4230 X1=IPOP(1)
      Y1=IPOP(1)
      X2=MACC1
      Y2=MACC2
      MACC3=SQRT((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1))
      IF(MACC3-20) 99,99,4231
 4231 MACC1=X1
      MACC2=Y1
      GO TO 4050
C
C
C     ROUTINE FOR CIRCLE DEFINED BY 3 POINTS ON CIRCUM
C
 4300 GO TO (4310,4320,4330,4340),LT
 4310 N=1
      GO TO 2
 4320 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 4330 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 4340 X3=MACC1
      Y3=MACC2
      X2=IPOP(1)
      Y2=IPOP(1)
      X1=IPOP(1)
      Y1=IPOP(1)
      C1=-(X1*X1+Y1*Y1)
      C2=-(X2*X2+Y2*Y2)
      C3=-(X3*X3+Y3*Y3)
      DET=X1*(Y2-Y3)-Y1*(X2-X3)+X2*Y3-X3*Y2
      IF(ABSF(DET)-5.)0099,4351,4351
 4351 D=(C1*(Y2-Y3)-Y1*(C2-C3)+C2*Y3-C3*Y2)/DET
      E=(X1*(C2-C3)-C1*(X2-X3)+X2*C3-X3*C2)/DET
      F=X1*(Y2*C3-C2*Y3)-Y1*(X2*C3-C2*X3)+C1*(X2*Y3-X3*Y2)
      F=F/DET
      MACC1=(-D/2.)
      MACC2=(-E/2.)
      MACC3=.5*SQRTF(D*D+E*E-4.*F)
      GO TO 4050
C
C
C     DRAW CIRCLE
```

```
C
 4050 IF(ID-1)3,4051,3
 4051 IF(LPLTSW)4052,4052,4053
 4052 CONTINUE
C            ROUTINE TO DISPLAY A CIRCLE
      GO TO 11
 4053 CONTINUE
C            ROUTINE TO PLOT A CIRCLE
      GO TO 11
C
C
C     CHOOSE ROUTINE FOR SPECIFIC ARC
C
 5000 GOTO(5100,5200,5300,5400,5500,5600),KSN
C
C
C     ROUTINE FOR ARC DEFINED BY CIRCLE AND TWO ANGLES
C
 5100 GO TO (5110,5120,5130,5140),LT
 5110 N=4
      GO TO 2
 5120 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=7
      GO TO 2
 5130 CALL STACK1(MACC1)
      N=7
      GO TO 2
 5140 IF(ABSF(MACC1)-3142) 5141,5141,5142
 5142 MACC1=MACC1-SIGNF(MACC1,3142)
 5141 MACC5=MACC1
      MACC1=IPOP(1)
      IF(ABSF(MACC1)-3142) 5143,5143,5144
 5144 MACC1=MACC1-SIGNF(MACC1,3142)
 5143 MACC4=MACC1
      MACC1=IPOP(1)
      MACC2=IPOP(1)
      MACC3=IPOP(1)
      GO TO 5050
C
C
C     ROUTINE FOR ARC DEFINED BY A CIRCLE AND TWO POINTS
C
 5200 GO TO (5210,5220,5230,5240),LT
 5210 N=4
      GO TO 2
 5220 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 5230 CALL STACK1(MACC2)
```

```
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 5240 X3=MACC2
      Y3=MACC1
      X2=IPOP(1)
      Y2=IPOP(1)
      X1=IPOP(1)
      Y1=IPOP(1)
      PI=3141.59
      DY=Y2-Y1
      DX=X2-X1
      IF(DX) 5242,5241,5242
 5241 IF(DY) 5243,99,5244
 5243 ALP=PI/2.
      GO TO 5245
 5244 ALP=-PI/2.
      GO TO 5245
 5242 ALP=ATANF(DY/DX)*1000.
      IF(DX) 5245,5245,5246
 5246 ALP=ALP-SIGNF(PI,ALP)
 5245 MACC4=ALP
      DY=Y1-Y3
      DX=X1-X3
      IF(DX) 5262,5261,5262
 5261 IF(DY) 5263,99,5264
 5263 ALP=PI/2.
      GO TO 5265
 5264 ALP=-PI/2.
      GO TO 5265
 5262 ALP=ATANF(DY/DX)*1000.
      IF(DX) 5265,5265,5266
 5266 ALP=ALP-SIGNF(PI,ALP)
 5265 MACC5=ALP
      GO TO 5050
C
C
C     ROUTINE FOR ARC DETER BY CIRCLE,PT,ANGLE
C
 5300 GO TO (5310,5320,5330,5340),LT
 5310 N=4
      GO TO 2
 5320 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 5330 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=7
      GO TO 2
 5340 A2=MACC1
      X1=IPOP(1)
```

```
      Y1=IPOP(1)
      X2=IPOP(1)
      Y2=IPOP(1)
      R=IPOP(1)
 5350 PI=3141.59
      DY=Y2-Y1
      DX=X2-X1
      IF(DX) 5342,5341,5342
 5341 IF(DY) 5343,99,5344
 5343 ALP=PI/2.
      GO TO 5345
 5344 ALP=-PI/2.
      GO TO 5345
 5342 ALP=ATANF(DY/DX)*1000.
      IF(DX) 5345,5345,5346
 5346 ALP=ALP-SIGNF(PI,ALP)
 5345 IF(A2) 5349,99,5347
 5349 A1=ALP+A2
      IF(ABSF(A1)-PI) 5351,5352,5352
 5352 A1=A1-SIGNF(2.*PI,A1)
 5351 A2=ALP
      GO TO 5348
 5347 A2=ALP+A2
      IF(ABSF(A1)-PI) 5353,5354,5354
 5354 A2=A2-SIGNF(2.*PI,A2)
 5353 A1=ALP
 5348 MACC1=X2
      MACC2=Y2
      MACC3=R
      MACC4=A1
      MACC5=A2
      GO TO 5050
C
C
C     ROUTINE FOR ARC DETER BY CIRCLE,PT,DIST
C
 5400 GO TO (5410,5420,5430,5440),LT
 5410 N=4
      GO TO 2
 5420 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 5430 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      CALL IDIST(NSI+LT,NPAR)
      IF(NPAR)5431,99,4
 5431 MACC1=-NPAR-60000
      LT=LT+1
 5440 D=MACC1
      X1=IPOP(1)
      Y1=IPOP(1)
```

```
      X2=IPOP(1)
      Y2=IPOP(1)
      R =IPOP(1)
 5441 IF(D-2.*R) 5442,5442,99
 5442 T=D/(2.*R)
      A2=2000.*ATANF(   T/SQRTF(1.-T*T))
      GO TO 5350
C
C
C     ROUTINE FOR ARC DETER BY CIRCLE,ANGLE DIST
C
 5500 GO TO (5510,5520,5530,5540),LT
 5510 N=4
      GO TO 2
 5520 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 5530 CALL STACK1(MACC1)
      CALL IDIST(NSI+LT,NPAR)
      IF(NPAR)5531,99,4
 5531 MACC1=-NPAR-60000
      LT=LT+1
 5540 D=MACC1
      A1=IPOP(1)
      X2=IPOP(1)
      Y2=IPOP(1)
      R =IPOP(1)
 5541 IF(D-2.*R) 5542,5542,99
 5542 T=D/(2.*R)
      A2=2000.*ATANF(   T/SQRTF(1.-T*T))
      GO TO 5351
C
C
C     ROUTINE FOR ARC DETER BY CIRCLE AND TWO DIST
C
 5600 GO TO (5610,5620,5630,5640),LT
 5610 N=4
      GO TO 2
 5620 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      CALL IDIST(NSI+LT,NPAR)
      IF(NPAR) 5621,99,4
 5621 MACC1=-NPAR-60000
      LT=LT+1
 5630 CALL STACK1(MACC1)
      CALL IDIST(NSI+LT,NPAR)
      IF(NPAR) 5631,99,4
 5631 MACC1=-NPAR-60000
      LT=LT+1
 5640 D2=MACC1
```

```
      D1=IPOP(1)
      X1=IPOP(1)
      Y1=IPOP(1)
      R =IPOP(1)
 5641 IF(D1-2.*R) 5642,5642,99
 5642 T=D1/(2.*R)
      A1=2000.*ATANF(T/SQRTF(1.-T*T))
      IF(D2-2.*R) 5643,5643,99
 5643 T=D2/(2.*R)
      A2=2000.*ATANF(T/SQRTF(1.-T*T))
      MACC4=A1
      MACC5=A2
      MACC1=X1
      MACC2=X2
      MACC3=R
      GO TO 5050
C
C
C     DRAW AN ARC
C
 5050 IF(ID-1) 3,5051,3
 5051 IF(LPLTSW)5052,5052,5053
 5052 CONTINUE
C      ROUTINE TO DISPLAY ARC
      GO TO 11
 5053 CONTINUE
C      ROUTINE TO PLOT ARC
      GO TO 11
C
C
C     CHOOSE ROUTINE FOR SPECIFIC DISTANCE
C
 6000 GOTO (6100,6200,6300,6400,6500,6600,6700),KSN
C
C
C     ROUTINE FOR DISTANCE DEFINED BY TWO PTS
C
 6100 GOTO (6110,6120,6130),LT
 6110 N=1
      GO TO 0002
 6120 CALL STACK1(MACC1)
      CALL STACK1(MACC2)
      N=1
      GO TO 0002
 6130 X2=MACC1
      Y2=MACC2
      Y1=IPOP(1)
      X1=IPOP(1)
      MACC1=SQRTF((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1))
      GO TO 6050
C
C
C     ROUTINE FOR HORZ DIST BETWEEN TWO PTS
```

```
C
 6200 GOTO (6210,6220,6230),LT
 6210 N=1
      GO TO 2
 6220 CALL STACK(MACC1)
      N=1
      GO TO 2
 6230 MACC1=ABSF(MACC1-IPOP(1))
      GO TO 6050
C
C
C     ROUTINE FOR VERT DIST BETWEEN TWO PTS
C
 6300 GOTO (6310,6320,6330),LT
 6310 N=1
      GO TO 2
 6320 CALL STACK(MACC2)
      N=1
      GO TO 2
 6330 MACC1= ABSF(MACC2-IPOP(1))
      GO TO 6050
C
C
C     ROUTINE FOR DIST DEFINED BY PT AND LINE
C
 6400 GOTO (6410,6420,6430),LT
 6410 N=1
      GO TO 2
 6420 CALL STACK(MACC1)
      CALL STACK(MACC2)
      N=2
      GO TO 2
 6430 X1=MACC1
      Y1=MACC2
      X2=MACC3
      Y2=MACC4
      Y3=IPOP(1)
      X3=IPOP(1)
      A=Y1-Y2
      B=X2-X1
      C=(Y2-Y1)*X1+(X1-X2)*Y1
      IF(ABSF(B)-5.)6433,6433,6434
 6433 MACC1=ABSF(X1-X3)
      GO TO 6050
 6434 IF(B)6431,6432,6432
 6432 MACC1=ABSF((A*X3+B*Y3+C)/SQRT(A*A+B*B))
      GO TO 6050
 6431 TX=X1
      TY=Y1
      X1=X2
      Y1=Y2
      X2=TX
      Y2=TY
```

```
      GOTO 6432
C
C
C     ROUTINE FOR DISTANCE DEFINED BY CIRCLE
C
 6500 GO TO (6510,6520),LT
 6510 N=4
      GO TO 2
 6520 MACC1=MACC3
      GO TO 6050
C
C
C     ROUTINE FOR DIST -- REAL NUMBER
C
 6600 GO TO( 6610,6620),LT
 6610 CALL IVAL(NSI+LT,NPAR)
      IF(NPAR) 5,99,6611
 6611 MACC1=NPAR-60000
      LT=LT+1
 6620 GO TO 6050
C
C
C     ROUTINE FOR BINARY OPERATION ON TWO DISTANCES
C
 6700 GO TO (6710,6720,6730,6740),LT
 6710 CALL IDIST(NSI+LT,NPAR)
      IF(NPAR) 6711,99,4
 6711 MACC1=-NPAR-60000
      LT=LT+1
 6720 CALL STACK1(MACC1)
      CALL IVAL(NSI+LT,NPAR)
      IF(NPAR) 5,99,6721
 6721 MACC1=NPAR-60000
      LT=LT+1
 6730 CALL STACK1(MACC1)
      CALL IDIST(NSI+LT,NPAR)
      IF(NPAR) 6731,99,4
 6731 MACC1=-NPAR-60000
      LT=LT+1
 6740 N=IPOP(1)/1000
      A=IPOP(1)
      B=MACC1
      GOTO(6741,6742,6743,6744,6745,6746,6747),N
 6741 MACC1=A+B
      GO TO 6050
 6742 MACC1=A-B
      GO TO 6050
 6743 MACC1=A*B
      GO TO 6050
 6744 MACC1=A/B
      GO TO 6050
 6745 MACC1=A**B
      GO TO 6050
```

```
 6746 MACC1=A*SQRT(ABSF(B))
      GO TO 6050
 6747 MACC1=A*ABSF(B)
      GO TO 6050
C
C
C      DRAW A DISTANCE
C
 6050 GO TO 3
C
C
C     CHOOSE ROUTINE FOR DEGREES
C
 7000 GOTO(7100,7200,7300),KSN
C
C
C     ROUTINE FOR DEGREES DETER BY TWO LINES
C
 7100 GO TO (7110,7120,7130),LT
 7110 N=2
      GO TO 2
 7120 CALL STACK1(MACC4)
      CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=2
      GO TO 2
 7130 X3=MACC1
      Y3=MACC2
      X4=MACC3
      Y4=MACC4
      X1=IPOP(1)
      Y1=IPOP(1)
      X2=IPOP(1)
      Y2=IPOP(1)
 7141 IF(ABSF(X1-X2)-20.) 7143,7143,7144
 7143 A1=3.14159/2.
      GO TO 7142
 7144 A1=ATANF((Y1-Y2)/(X1-X2))
 7142 IF(ABSF(X3-X4)-20.) 7146,7146,7147
 7146 A2=3.14159/2.
      GO TO 7148
 7147 A2=ATANF((Y3-Y4)/(X3-X4))
 7148 MACC1=ABSF(A1-A2)
      GO TO 7050
C
C
C     ROUTINE FOR DEGREES DETER BY 3 PTS
C
 7200 GO TO (7210,7220,7230,7240),LT
 7210 N=1
      GO TO 2
 7220 CALL STACK1(MACC2)
```

```
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 7230 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 2
 7240 X3=MACC1
      Y3=MACC2
      X2=IPOP(1)
      Y2=IPOP(1)
      X1=IPOP(1)
      Y1=IPOP(1)
 7241 X4=X3
      Y4=Y3
      X3=X2
      Y3=Y2
      GO TO 7141
C
C
C            ANGLE DETERMINED BY A REAL NUMBER
C
 7300 GO TO (7310,7320),LT
 7310 CALL IVAL(NSI+LT,NPAR)
      IF(NPAR) 5,99,7311
 7311 MACC1=NPAR-60000
      LT=LT+1
 7320 GO TO 7050
C
C
C       DRAW AN ANGLE
C
 7050 GO TO 3
C
       END



      SUBROUTINE ICHASE (M,K,NPAR)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
      LL=K*1000
      LU=LL+1000
      I=LPT2
      J=L(M)
   13 IF(J) 2,1,1
    1 IF(J-30000) 15,16,16
   15 N=L(J)
      IF(K-8) 18,19,18
   19 IF(N/1000-6) 4,20,4
```

```
   20 NPAR=-J
      RETURN
   18 IF( N  -LL) 4,3,3
    3 IF(LU- N  ) 4,4,5
    4 IF( N ) 6,7,7
    6 NPAR=-3
      RETURN
    7 IF(N-1000) 8,9,9
    8 NPAR =-9
      RETURN
    9 IF(N-3000) 12,8,10
   10 IF(N-4000)  8,11,11
   11 IF(N-8000) 12,8,8
   12 NPAR  =-N/1000
      RETURN
    5 NPAR=J
      RETURN
    2 IF(I) 8,8,14
   14 J=LST2(I)-J
      J=L(J)
      I=I-1
      GO TO 13
   16 IF(K-8) 21,5,17
   21 IF(K-6) 17,20,17
   17 NPAR=-8
      RETURN
      END


      FUNCTION IDENTY(J)
C     IDENTIFIES TYPE OF WORD
      END


      SUBROUTINE IDIST ( M,NPAR )
      CALL ICHASE ( M,6,NPAR)
      IF(NPAR) 1,1,2
    1 IF(NPAR+10) 2,3,3
    3 NPAR=0
    2 RETURN
      END


      FUNCTION INTGER(J,N)
C     CONVERTS WORD TO INTEGER VALUE
      END


      FUNCTION IPOP(N)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
```

```
      IF(N-1) 20,20,21
   20 IPOP=LST1(LPT1)
      LPT1=LPT1-1
      RETURN
   21 IPOP=LST2(LPT2)
      LPT2=LPT2-1
      RETURN
      END


      SUBROUTINE IVAL(M,NPAR)
      CALL ICHASE(M,8,NPAR)
      IF(NPAR)1,1,2
    1 IF(NPAR+10) 2,2,3
    3 NPAR=0
    2 RETURN
      END


      FUNCTION LALLOC(N)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
      IF(N-4) 1,1,2
    1 N=4
    2 IF(LALLPT+N-300) 3,3,4
    3 LALLOC=LALLPT
      LALLPT=LALLPT+N
      RETURN
    4 PRINT 5
    5 FORMAT(1H ,40HMAIN MEMORY ARRAY SIZE EXCEEDED          )
      STOP 01
      END


      FUNCTION LSRCHN(N)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
      IF(NTLPT) 1,1,2
    1 LSRCHN=0
      RETURN
    2 DO 3 I=1,NTLPT
      IF(N-NTL(I,1))   3,4,3
    3 CONTINUE
      GO TO 1
    4 LSRCHN=NTL(I,2)
      RETURN
      END
```

```
      FUNCTION LSRCHP(N)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
      IF(NTLPT) 1,1,2
    1 LSRCHP=0
      RETURN
    2 DO 3 I=1,NTLPT
      IF(N-NTL(I,2))   3,4,3
    3 CONTINUE
      GO TO 1
    4 LSRCHP=NTL(I,1)
      RETURN
      END


      SUBROUTINE MAN ( X,Y, X1,Y1)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
      X1=S+A*X+B*Y
      Y1=T+C*X+D*Y
      RETURN
      END


      SUBROUTINE RDLN(K)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
      READ  90, LIN(1,1), LIN(1,2), LIN(2,1), LIN(2,2),
     1   (LIN(I,1),I=3,9),LC
   90 FORMAT(A4,1X,A6,1X,A6,2X,I1,7(1X,A6   ),1X,I1)
      IF(LC) 22,22,1
   22 LIN(10,1)=6H
      GO TO 2
    1 READ  91,( LIN(I,1),I=10,16),LC
   91 FORMAT(22X,7(A6,1X),I1)
      IF(LC) 23,23,3
   23 LIN(17,1)=6H
      GO TO 2
    3 READ  91,( LIN(I,1),I=17,23),LC
      IF(LC)24,24,4
   24 LIN(24,1)=6H
      GO TO 2
    4 READ  91,( LIN(I,1),I=24,27),LC,LC,LC,LC
      IF(LC) 2,2,6
    6 K=1
      RETURN
    2 N=IDENTY(LIN(1,2))
```

```
      IF(N) 5,6,6
    5 DO 7 I=3,27
      N=IDENTY(LIN(I,1))
      IF(N+1) 25,15,8
    8 IF(N) 13,14,13
   13 IF(N-1) 15,14,6
   14 LC=INTGER(LIN(I,1),N)
      LIN(I,1)=LC
   15 LIN(I,2)=N
    7 CONTINUE
      I=27
      GO TO 9
   25 I=I-1
    9 IF(LIN(2,2)) 10,10,11
   10 LIN(2,2) = I-2
      GO TO 12
   11 LIN(7,1)=I-2
   12 K=0
      RETURN
      END


      SUBROUTINE STACK1(K)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
      LPT1=LPT1+1
      IF(LPT1-100) 2,2,1
    1 PRINT 10010
      STOP 04
    2 LST1(LPT1)=K
      RETURN
10010 FORMAT(1H ,40HSTACK 1 SIZE EXCEEDED                    )
      END


      SUBROUTINE STACK2(K)
      DIMENSION L(300),LIN(27,2),LST1(100),LST2(40)
      DIMENSION NTL(100,2),LBLK(80)
      COMMON LALLPT,NTLPT,NTL,LBLK,LST1,LPT1,LPT2,L,LST2
      COMMON A,B,C,D,S,T,LIN
      LPT2=LPT2+1
      IF(LPT2-40) 2,2,1
    1 PRINT 10011
      STOP 05
    2 LST2(LPT2)=K
      RETURN
10011 FORMAT(1H ,40HSTACK 2 SIZE EXCEEDED                    )
      END
```

Notes: The numbers in the right hand margin refer to blocks on the block diagrams.

The subroutine IDENTY identifies a word in A6 format as: blank: -2; a name: -1; a number: 0; a parameter: 1.

The subroutine INTGER converts numbers and integers from A6 to I format.

All communications with the user are given in the form of PRINT statements.

Display and plot routines are necessary for each of the types of BCU's.

V. G.D.L. - 2

a) Introduction

As progress on the first version of the language progressed it became very clear that the decision to partially assemble the language statements into the block form outlined was leading to a good deal of artificiality in the language implementation, and Handler commands. Particularly changing, adding and deleting were not terribly easy (c.f. the Dummy statement), and hampering restrictions were placed on the user (for example, names within a figure were globally, not locally, defined).

The problem of garbage collection became important, as the deletion method employed left all structures in memory. Garbage collection was very difficult indeed in the GDL-1 system as the movement of anything required the changing of all pointers to the thing moved; as no record of references was intrinsic to the data structure scheme, it proved nearly impossible to locate these pointers.

The trouble stemmed from the loss of information available once names were replaced by pointers. A pointer is location dependent, while a name is not. Hence, once the partial compilation had been completed the structure became totally location dependent.

To meet these drawbacks, a second form of the language was designed and implemented. It is more interpretive in nature and a call-by-name scheme is used throughout.

b) Basic Structures

The basic structure of GDL-2 is the line. Each line is generated by a single typed input line. Lines are stored as a file. There are three types of lines; figure (FL), subfigure (SL), and basic construction unit (BCUL). A figure is defined as those SL and BCUL lines between a FL line indicating the figure and the FL line beginning the next figure. The format of each line is essentially that of the typed input line:

A) Figure lines (FL): These have three words. The first is a name, the second a 0, the third a length indicator set = 3. These are used to indicate the beginning of new figures. The figure whose name is in word 1 of the FL is that set of SL and BCUL lines following that FL and preceding the next FL. A figure name table is kept giving pointers to each FL currently in the file.

B) Subfigure lines (SL): These have n + 9 words where n is the number of parameters in a subfigure call within a figure. Such a line corresponds exactly to the PAM - block and the pointer in a figure block to it in GDL-1.

The words are:
1. name
2. -1
3. a length indicator = n + 9
4. the name of the subfigure to be drawn
5. x_0
6. y_0
7. θ
8. E

9. through n + 8. n parameters, each a name, or value, or a parameter.

n + 9. a length indicator = n + 9.

C) Basic construction unit lines (BCUL): These correspond exactly to BCU blocks in GDL-1. They have n + 4 words, where n is the number of other pieces of information required to define this basic (i.e., the number of pointers in the BCU in GDL-1).

They are: 1. name

2. k x 10000 + xy00 where k is 1 if the BL is to be drawn, and 0 otherwise; x is the type digit, and y is the kind digit.

3. length indicator set = n + 4.

4. through n + 3. the n pieces of information defining the BCUL.

n + 4. length indicator set = n + 4.

c) Handler Commands and Effects

GDL-2 has only two basic modes of operation. One is the FILE CREATION mode and the other is the PLOT mode.

Initially, or after a plot has been made, control is immediately returned to the file creation mode; this program is referred to as the "Handler".

The file is created as a list of lines. Initially the file consists of only an End-Of-File (EOF) line. An arrow is set pointing above the top of the file, namely to the EOF - line initially. This arrow points between lines in the file at all times.

The following commands can be given to the handler; examples are given in Figure 23:

1. Figure: this sets up a figure line. The name given is entered in the name table and the line inserted below the arrow. The arrow is reset pointing below this line.

2. Subfig, Point, ..., Angle: these set up their corresponding type of line, as in the figure command.

3. Locate: this command sets the arrow above:

a) a figure line: the figure name given is looked up in the table, and the arrow set accordingly.

b) the first line in the file: the word FIRST calls this option.

c) the EOF line: the word NEW calls the option.

4. Skip: this skips the arrow over a number of lines. The options are:

a) n > 0. This skips the arrow over n lines in a downward direction. The skip will only reach the end of the figure. If n lines have not been skipped when this happens, a message is given and the arrow set above the figure - line following or the EOF line.

b) n < 0. This skips the arrow over n lines in an upward direction. The skip will only reach the figure line for the figure in which the arrow is before the command. If n lines have not been skipped when the figure line is encountered a message is given and the arrow set above the figure line in question.

c) n = 0, or blank. Default condition: n = 1.

5. Delete: This deletes a number of lines. The options are:

```
1--4 6---11 13--18 21 23--28 30--35 37--41 44--49 51--56 . . .

          .
          LOCATE    SQUARE
          SKIP        5          Skips to line SQUAB
          DELETE      2          Deletes lines SQUAB, and SQUBC
          SKIP      - 1          Sets pointer above line SQUD
          .

          .
          LOCATE    SQUARE
          DELETE      0          Deletes all of SQUARE
          .

          .
          LOCATE    SQUARE
          PRINT     FIGURE       Prints all of SQUARE
          PRINT       3          Prints lines SQUARE, SQUA and SQUB
          .

          .
          DISPLA    SQUARE   0       0       0       0
          PLOT      SQUARE   0       0       0       1
          FINISH
```

Figure 23. Some Examples of GDL-2 Handler Commands
(cf. Figure 2 for Lines Defining Figure SQUARE)

a) n > 0. Deletes the n lines following the arrow. However, if a figure line is encountered first, a message is given and no deleting is done.

b) n = 0. Deletes a whole figure. The arrow must be set above a figure line. If this is not the situation, a message is given and no deletion takes place. If it is on a figure line, the whole figure it begins is deleted.

c) n is blank. Default condition : n = 1.

Any delete command causing a figure line to be deleted also causes that figure name to be deleted from the name table.

6. Print: This causes portions of the file to be printed for inspection.

The options are:

a) ALL: Prints the complete file.

b) FIGURE: Prints out a complete figure.

The arrow must be set above a figure line. Otherwise, an error message is given and no printing is done. If the arrow is set correctly, the figure line and all the lines composing the figure are printed.

c) n > 0. Prints out the n lines immediately following the arrow regardless of type. However, if the EOF line is encountered before n lines have been printed, a message is given and printing is terminated.

d) n = 0, or blank. Default condition: n = 1

All print commands leave the position of the arrow unchanged.

7. Display: This causes the mode to change to display mode. The figure given is displayed.

8. Plot: This causes the mode to change to plot mode. The figure given is plotted on the CalComp Plotter.

9. Finish: This calls the system monitor.

The various Handler commands can be given in any order whatsoever. Errors in the usage of them cause error messages, some of which have been indicated in the discussion of the commands.

d) Plotting, Displaying, and Representation in Fortran

The techniques of plotting and displaying used in GDL-2 are almost identical to those used in GDL-1. The PLOT and DISPLAY commands, cause control to be transferred to the drawing program; the program is then considered to be in the PLOT or DISPLAY mode. The actions taken by the drawing program are the same for both the PLOT and DISPLAY commands; the only difference is that when the description of a line or circle segment has been determined and is to be drawn, a PLOT command calls x-y plotter subroutines, while a DISPLAY command calls CRT display subroutines.

Initially, the drawing program sets up the x_0, y_0, Θ, and E parameters in their vector format $\begin{pmatrix} S \\ T \end{pmatrix}$ and $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$, forms a partial line of parameters for the figure to be drawn, stacking its location in LST2 - the parameter stack, and locates the FIGURE - line of that figure.

Control is transferred to a routine which steps through each line in the figure to be drawn, checking the type-kind indicator

for over 10000. This routine, controls the calling of the correct routines to calculate the descriptions of each of the BCUL's which is to be drawn.

The selection of the correct calculating routine is done on the basis of the type and kind of line encountered. This information is stored in each line as has been discussed. Control is then passed to the code for that type and kind of line.

This code in turn will often have to call still further blocks of code to calculate the BCU's upon which it depends for its description.

As in GDL-1, a push-down stack is used to allow arbitrarily deep nesting of BCU's. Suppose, for instance, that a BCUL defining a line segment by its end points is to be drawn. The type-kind indicator will then be 12100 = 10,000 (as it is to be drawn) + 2100 (a line segment - 2 - of kind 1). Control is transferred through a 'computed - GO - TO' statement to the block of code starting with statement number 2100. This code calls a subroutine, called ICHASE, to locate the BCUL represented by the name in word 1. If there is any trouble in locating this, or if the BCUL represented by word 1 does not represent a point an error return is given by ICHASE. This error return causes an error message to be given, the drawing program to be terminated, and the program to be returned to the FILE CREATION mode. ICHASE locates the desired BCUL by searching upward through the file until it is encountered, or until the FIGURE - line for the figure being drawn is encountered. This latter is an error condition. It should be noted that this procedure demands predefinition of all

BCU names used within a figure.

If ICHASE is successful, the location of the current BCUL is stored in the stack, along with an indicator (called LT in FORTRAN) of which word in the current BCUL is being analyzed, in this case word 1. The locationof the new BCUL represented by word 1 as determined by ICHASE becomes the current BCUL and LT becomes 1. This procedure is called 'going down a stage'.

The same procedure may be repeated in this new BCUL. At some point the program will be able to calculate the co-ordinates of a point. These will be stored in an accumulator (MACC n, where $1 \leq n \leq 5$) and the program returns up a stage. Suppose that the point represented by word 1 in our example was represented by its co-ordinate. The code for this type and kind of BCUL would simply load the accumulator with these values.

Control is then returned to the routine to 'go up a stage'. This routine obtains the old LT and location of the old BCUL. Analyzing the type-kind indication in the BCUL the program again goes to the code for a 2100 - line. Now LT is updated, and the second word is examined. Before going down a stage for the second word, the values of the first are stacked from the accumulator. Then ICHASE is called on the second word and the processing continues, searching down as many stages as is necessary, stacking what needs to be saved. This can be thought of as doing a prefix scan on the tree in which each BCUL represents a node and each word used with a name a branch.

Throughout this procedure an indicator (ID) is decremented by 1 each time the program goes down a stage, and augmented by 1 each

time it goes up a stage. It is initially set at 1 by the routine which steps through the figure to be drawn. When it again becomes 1, the program knows that it should draw the BCU represented by the current BCUL. Instead of returning values to the accumulator, it calls the plotter or CRT routines, using the values calculated, as manipulated according to the current values of A, B, C, D, S, and T.

After drawing, control is returned to the routine which steps through the figure. This searches for the next BCUL to be calculated and drawn.

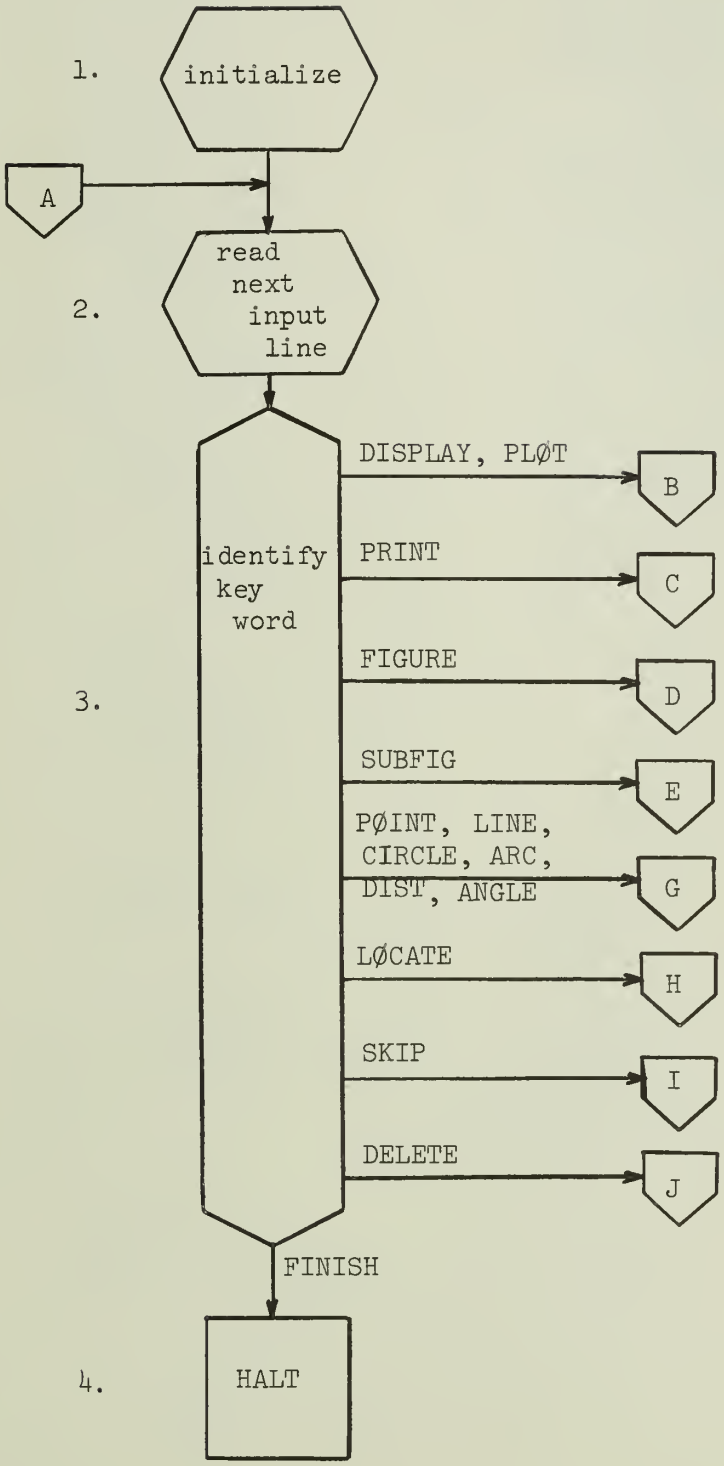
If this routine encounters a line which is a SUBFIG - line, it recognizes the necessity to 'go down a level'. It calls a routine to do this. This routine stores the current values of A, B, C, D, S, and T, calculates new values for A, B, C, D, S, T based on the old values and the values of x_0, y_0, θ, E in this BCUL, stacks in the parameter stack the location of the parameter list of this BCUL (word 6), locates the subfigure to be drawn, stacks the location of this BCUL for later return, and returns control of the routine which steps through this new figure.

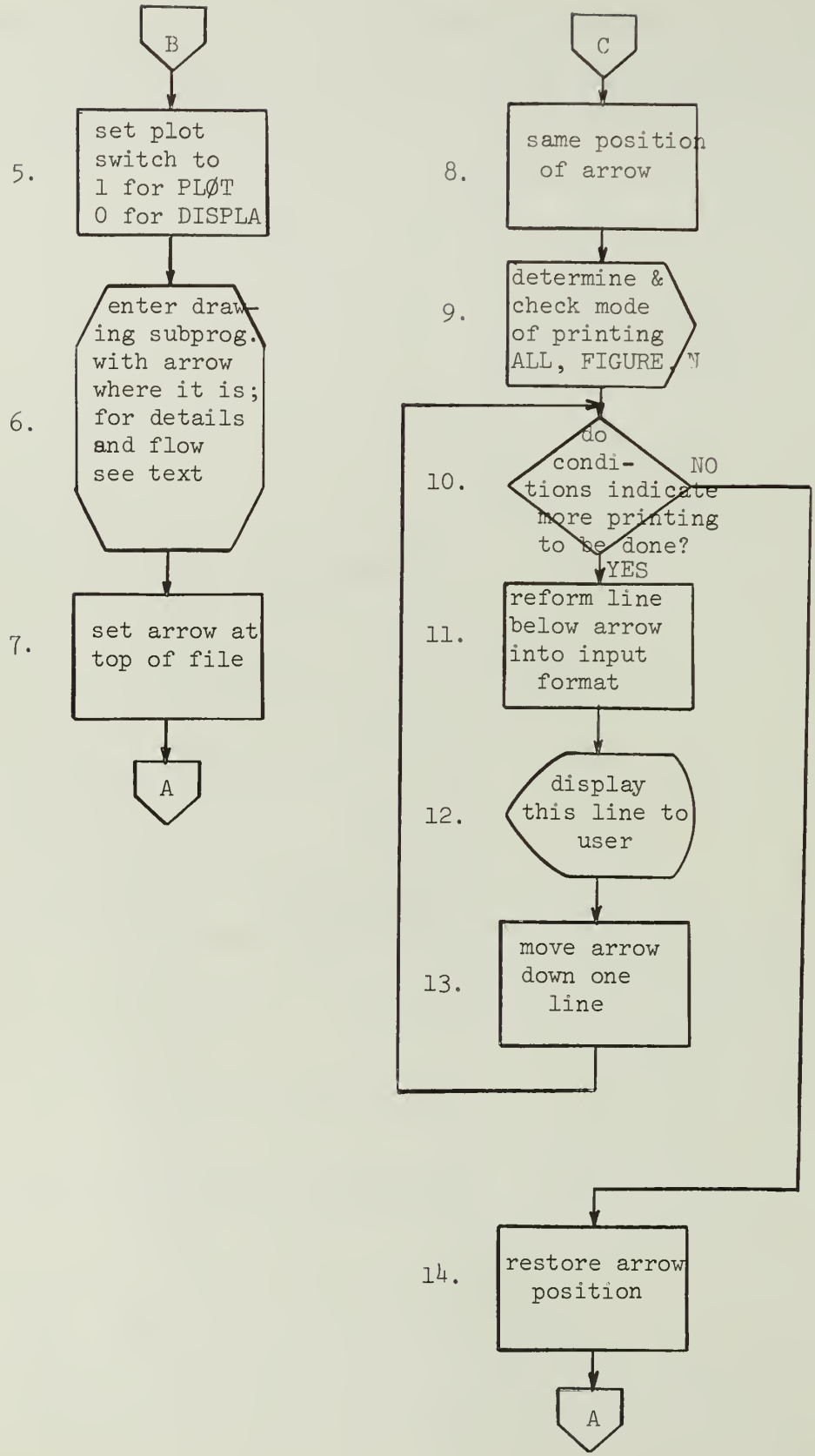
The parameter stack allows ICHASE to search back up through various levels of figure to locate any parameter encountered while calculating BCU's.

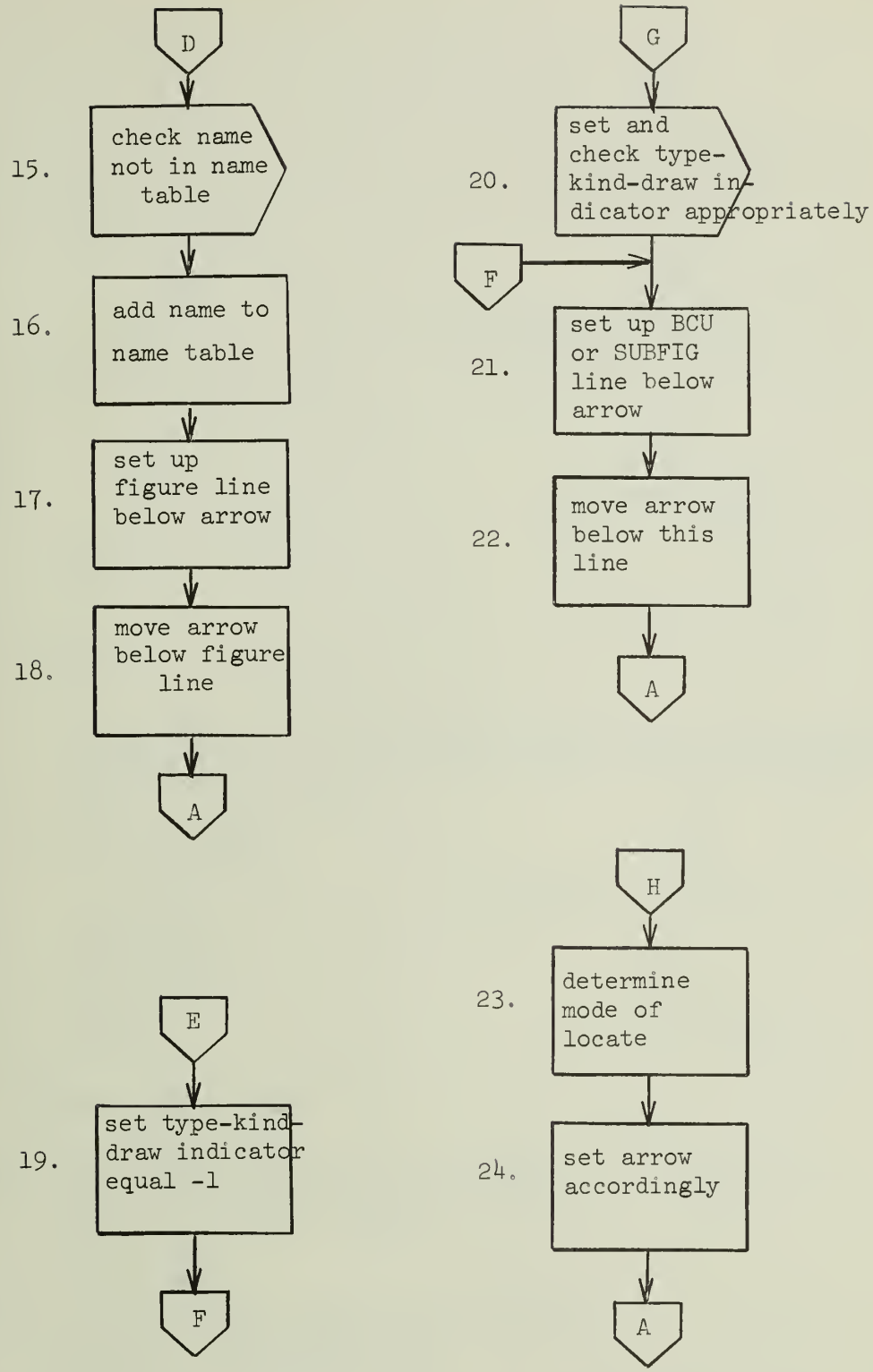
When the routine which steps through a figure encounters the FIGURE - line of the following figure, it recognizes the need to 'return up a level'. The old values of A, B, C, D, S, T are removed from the stack, as is the location of the BCUL which caused the program to go down the level; control is returned to the stepping program at the BCUL following this one. A check is made on whether the

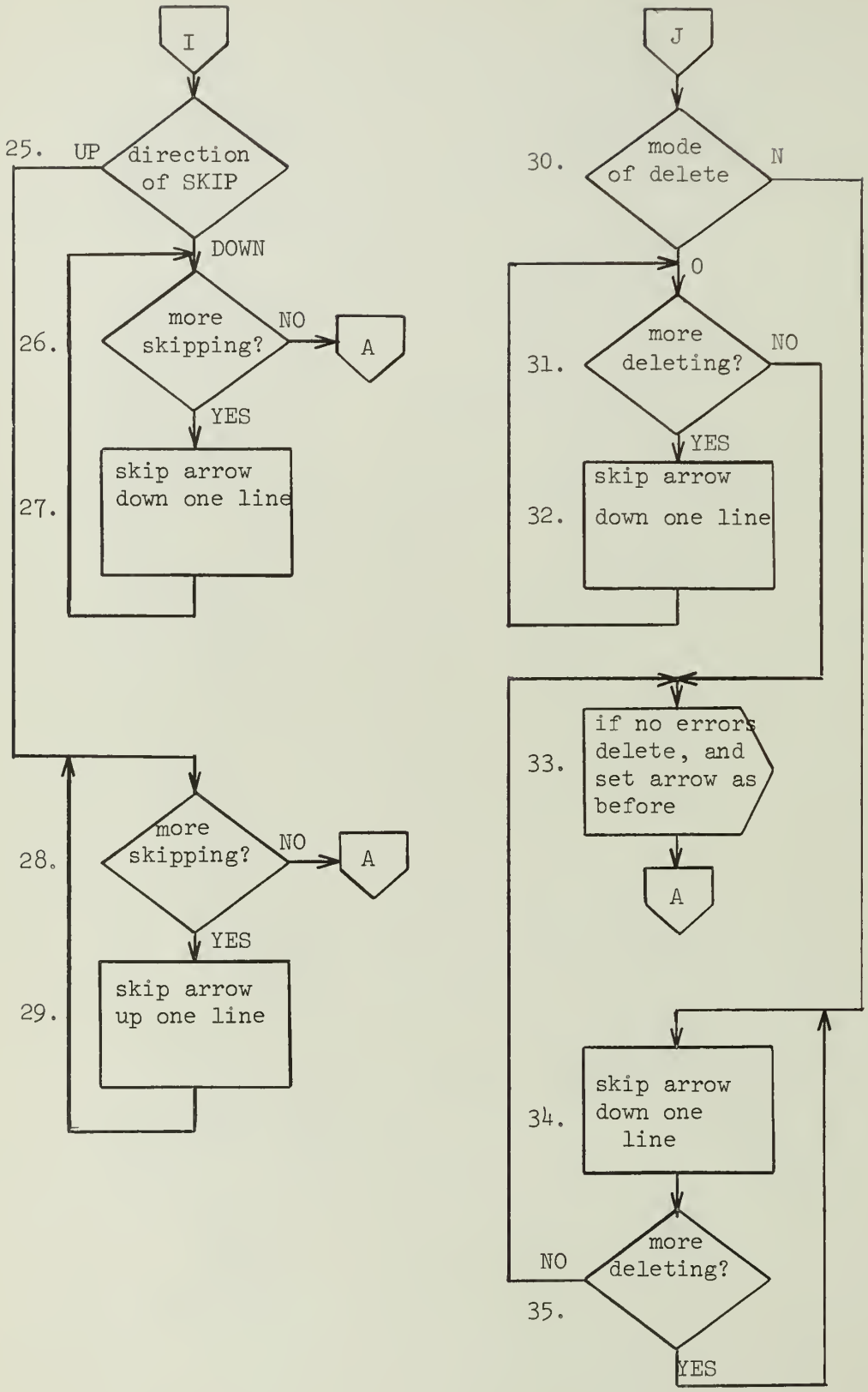
figure just finished is the main figure being drawn. If this is the case, control is returned to the Handler and the program reverts to FILE CREATION mode.

e) Block Diagrams




B
5.
set plot switch to 1 for PLØT 0 for DISPLA
6.
enter drawing subprog. with arrow where it is; for details and flow see text
7.
set arrow at top of file
A
C
8.
same position of arrow
9.
determine & check mode of printing ALL, FIGURE, N
10.
do conditions indicate more printing to be done?
NO
YES
11.
reform line below arrow into input format
12.
display this line to user
13.
move arrow down one line
14.
restore arrow position
A


D
15.
check name not in name table
16.
add name to name table
17.
set up figure line below arrow
18.
move arrow below figure line
A
G
20.
set and check type-kind-draw indicator appropriately
F
21.
set up BCU or SUBFIG line below arrow
22.
move arrow below this line
A
E
19.
set type-kind-draw indicator equal -1
F
H
23.
determine mode of locate
24.
set arrow accordingly
A


I
25.
UP
direction of SKIP
DOWN
more skipping?
NO
A
26.
YES
skip arrow down one line
27.
more skipping?
NO
A
28.
YES
skip arrow up one line
29.
J
30.
mode of delete
N
O
more deleting?
NO
31.
YES
skip arrow down one line
32.
if no errors delete, and set arrow as before
33.
A
skip arrow down one line
34.
more deleting?
NO
35.
YES

f) Program Listing

Refer to the notes at the end of Section IV-f.

```
C      PROGRAM  GDL-2
C
C
C     HANDLER
C
C
       DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
       COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
       COMMON A,B,C,D,S,T
C
C  INITIALIZATION.
C
       NTLPT=0                                                    1
       LALOPT=22
       PRINT  1
     1 FORMAT(1H1)
       L(22)=0
       LA=22
C
C  READ THE NEXT LINE
C
 9200 READ 9201,(LIN(I),I=1,11),LC                                2
 9201 FORMAT(A4,2(1X,A6),2X,I1,7(1X,A6),1X,I1)
      IF(LC) 9202,9202,9203
 9202 LIN(12)=6H
      GO TO 9210
 9203 READ   9204,( LIN(I),I=12,18),LC
 9204 FORMAT(22X,7(A6,1X),I1)
      IF(LC) 9205,9205,9206
 9205 LIN(19)=6H
      GO TO 9210
 9206 READ   9204,( LIN(I),I=19,25),LC
      IF(LC) 9207,9207,9208
 9207 LIN(26)=6H
      GO TO 9210
```

```
 9208 READ   9209,( LIN(I),I=26,29),LC
 9209 FORMAT(22X,4(A6,1X),21X,I1)
      IF(LC) 9210,9210,9226
 9226 PRINT 10015
      GO TO 9200
C
C TESTS
C
C
 9210 IF(LIN(3)-6HFINISH) 9211,9250,9211                              3
 9211 IF(LIN(3)-6HDISPLA) 9212,9260,9212
 9212 IF(LIN(3)-6HPLOT  ) 9213,9270,9213
 9213 IF(LIN(3)-6HPRINT ) 9214,9280,9214
 9214 IF(LIN(3)-6HFIGURE) 9215,9330,9215
 9215 IF(LIN(3)-6HPOINT ) 9216,9350,9216
 9216 IF(LIN(3)-6HLINE  ) 9217,9360,9217
 9217 IF(LIN(3)-6HCIRCLE) 9218,9370,9218
 9218 IF(LIN(3)-6HARC   ) 9219,9380,9219
 9219 IF(LIN(3)-6HDIST  ) 9220,9390,9220
 9220 IF(LIN(3)-6HANGLE ) 9221,9400,9221
 9221 IF(LIN(3)-6HSUBFIG) 9222,9410,9222
 9222 IF(LIN(3)-6HLOCATE) 9223,9420,9223
 9223 IF(LIN(3)-6HSKIP  ) 9224,9430,9224
 9224 IF(LIN(3)-6HDELETE) 9225,9450,9225
 9225 PRINT 10016
      GO TO 9200
C
C  FINISH ORDER
C
 9250 STOP 10                                                         4
C
C  DISPLAY COMMAND
C
 9260 LPLTSW=0                                                        5
      GO TO 9000                                                      6
C          I.E. GO TO PLOTTING SUBPROGRAM
 9999 LA=22                                                           7
      GO TO 9200
C
C  PLOT COMMAND
C
 9270 LPLTSW=1                                                        5
      GO TO 9000                                                      6
C
C PRINT COMMAND
C
 9280 NSI=LA                                                          8
      IF(LIN(5)-6HALL   ) 9281,9289,9281                              9
 9281 IF(LIN(5)-6HFIGURE) 9282,9286,9282
 9282 K=IDENTY(LIN(5))
      IF(K) 9300,9284,9283
 9300 IF(K+2) 9283,9299,9283
 9283 PRINT 10017
```

```
      GO TO 9219
9284  N=INTGER(LIN(5),K)/1000
      IF(N) 9283,9299,9285
9299  N=1
9285  LS=3
      GO TO 9290
9286  IF(L(LA)) 9287,9288,9287
9287  PRINT 10018
      GO TO 9319
9288  LS=2
      GO TO 9290
9289  LA=22
      LS=1
9290  IF(LA-LALOPT) 9291,9316,9316                                    10
9291  IF(L(LA)) 9292,9293,9295
9292  LIN(3)=6HSUBFIG                                                 11
9294  LIN(1)=6H
      KPSW=0
      GO TO 9307
9293  LIN(3)=6HFIGURE
      GO TO 9294
9295  K=L(LA)/10000
      KPSW=1
      IF(K) 9296,9296,9297
9297  LIN(1)=6HDRAW
      GO TO 9298
9296  LIN(1)=6H
9298  M=L(LA)-K*10000
      K=M/1000
      M=(M-K*1000)/100
      LIN(4)=M
      GO TO (9301,9302,9319,9303,9304,9305,9306),K
9301  LIN(3)=6HPOINT
      GO TO 9307
9302  LIN(3)=6HLINE
      GO TO 9307
9303  LIN(3)=6HCIRCLE
      GO TO 9307
9304  LIN(3)=6HARC
      GO TO 9307
9305  LIN(3)=6HDIST
      GO TO 9307
9306  LIN(3)=6HANGLE
9307  LIN(2)=L(LA-1)
      K=L(LA+1)-4
      DO 9308 I=1,25
      IF(I-K) 9310,9310,9309
9309  LIN(I+4)=6H
      GO TO 9311
9310  LB=LA+I+1
9308  LIN(I+4)=L(LB)
      I=25
9311  IF(KPSW) 9320,9320,9321
```

```
 9320 PRINT 9322,( LIN(J),J=1,3),( LIN(J+4),J=1,I)          12
 9322 FORMAT(1H ,A4,1X,2(A6,1X),3X,7(A6,1X)/(23X,7(A6,1X)))
      GO TO 9313
 9321 PRINT  9312,( LIN(J),J=1,3),LIN(4),( LIN(J+4),J=1,I)
 9312 FORMAT(1H ,A4,1X,2(A6,1X),I2,1X,7(A6,1X)/
     1          (23X,7(A6,1X)))
 9313 LA=LA+L(LA+1)                                         13
      GO TO (9290,9314,9315),LS
 9314 IF(L(LA)) 9290,9319,9290
 9315 N=N-1
      IF(N) 9290,9319,9290                                  10
 9316 GO TO(9319,9317,9318),LS
 9317 PRINT 10019
      GO TO 9319
 9318 PRINT 10020                                           14
 9319 LA=NSI
      GO TO 9200
C
C FIGURE COMMAND
C
 9330 N=IDENTY(LIN(2))                                      15
      IF(N+1) 9331,9332,9331
 9331 PRINT 10021
      GO TO 9200
 9332 N=LSRCHN(LIN(2))
      IF(N) 9334,9334,9333
 9333 PRINT 10022
      GO TO 9200
 9334 N=LALOCT(LA,3)                                        17
      IF(N) 9335,9335,9336
 9335 PRINT 10023
      GO TO 9200
 9336 CALL NAMADD(LIN(2),LA)                                16
      L(LA-1)=LIN(2)                                        17
      L(LA)=0
      L(LA+1)=3
      LA=LA+3                                               18
      GO TO 9200
C
C  COMMON SET UP ROUTINE FOR B.C.U. LINES
C
 9340 IF(LIN(4)-LC) 9348,9348,9349                          20
 9348 IF(LIN(4)) 9349,9349,9339
 9349 PRINT 10024
      GO TO 9200
 9339 N=K*1000+LIN(4)*100
      IF(LIN(1)-6HDRAW  ) 9341,9342,9341
 9342 N=N+10000
 9341 DO 9343 I=5,29                                        21
      M=IDENTY(LIN(I))
      IF(M+2) 9343,9345,9343
 9343 CONTINUE
      I=30
```

```
 9345 K1=I-1
      M=LALOCT(LA,K1)
      IF(M) 9351,9351,9352
 9351 PRINT 10023
      GO TO 9200
 9352 L(LA-1)=LIN(2)
      L(LA)=N
      L(LA+1)=K1
      LB=LA+K1-2
      L(LB)=K1
      DO 9347 I=5,K1
      LB=LA+I-3
 9347 L(LB)=LIN(I)
      LA=LA+K1                                                    22
      GO TO 9200
C
C POINT COMMAND
C
 9350 K=1                                                         20
      LC=4
      GO TO 9340
C
C LINE COMMAND
C
 9360 K=2                                                         20
      LC=2
      GO TO 9340
C
C  CIRCLE COMMAND
C
 9370 K=4                                                         20
      LC=3
      GO TO 9340
C
C ARC COMMAND
C
 9380 K=5                                                         20
      LC=6
      GO TO 9340
C
C DISTANCE COMMAND
C
 9390 K=6                                                         20
      LC=7
      GO TO 9340
C
C DEGREE COMMAND
C
 9400 K=7                                                         20
      LC=3
      GO TO 9340
C
C SUBFIGURE COMMAND
```

```
C
 9410 N=-1                                                          19
      GO TO 9341
C
C LOCATE COMMAND
C
 9420 IF(LIN(5)-6HFIRST ) 9421,9422,9421                            23
 9421 IF(LIN(5)-6HNEW   ) 9423,9424,9423
 9423 N=IDENTY(LIN(5))
      IF(N+1) 9425,9426,9425
 9425 PRINT 10025
      GO TO 9200
 9426 N=LSRCHN(LIN(5))                                              24
      IF(N) 9427,9427,9428
 9427 PRINT 10026
      GO TO 9200
 9428 LA=N
      GO TO 9200
 9422 LA=22
      GO TO 9200
 9424 LA=LALOPT
      GO TO 9200
C
C SKIP COMMAND
C
 9430 N=IDENTY(LIN(5))
      IF(N) 9442,9443,9444
 9442 IF(N+2) 9444,9432,9444
 9444 PRINT 10036
      GO TO 9200
 9443 N=INTGER(LIN(5),N)/1000
      IF(N) 9431,9432,9433                                          25
 9432 N=1
 9433 IF(LA-LALOPT) 9434,9435,9435                                  26
 9435 PRINT 10027
      GO TO 9200
 9434 LA=L(LA+1)+LA                                                 27
      N=N-1
      IF(N) 9436,9200,9436                                          26
 9436 IF(L(LA)) 9433,9437,9433
 9437 PRINT 10028
      GO TO 9200
 9431 IF(LA-22) 9438,9438,9439                                      28
 9438 PRINT 10029
      GO TO 9200
 9439 IF(L(LA)) 9440,9441,9440
 9441 PRINT 10030
      GO TO 9200
 9440 LA=LA-L(LA-2)                                                 29
      N=N+1                                                         28
      IF(N) 9431,9200,9431
C
C DELETE COMMAND
```

```
C
 9450 LS=LA
      N=IDENTY(LIN(5))
      IF(N) 9462,9457,9456
 9462 IF(N+2) 9456,9463,9456
 9463 N=1
      GO TO 9453
 9456 PRINT 10033
      GO TO 9200
 9457 N=INTGER(LIN(5),N)/1000
      LC=0
      IF(N) 9451,9452,9453                                          30
 9451 PRINT 10031
      GO TO 9200
 9452 IF(L(LA)) 9454,9455,9454                                      31
 9454 PRINT 10032
      GO TO 9200
 9455 IF(LA-LALOPT) 9458,9459,9458
 9459 PRINT 10034
      GO TO 9200
 9458 LS=LS+L(LS+1)                                                 32
      LC=LC+L(LS-2)
      IF(L(LS)) 9460,9458,9460                                      31
 9453 LS=LS+L(LS+1)                                                 34
      LC=LC+L(LS-2)
      N=N-1
      IF(N) 9461,9460,9461                                          35
 9461 IF(L(LS)) 9453,9464,9453
 9464 PRINT 10035
      GO TO 9200
 9460 N=LALOCT(LS,LC)                                               33
      GO TO 9200
C
C
C          HANDLER ERROR MESSAGES
C
10001 FORMAT(1H ,40HFIGURE NAME IN DISPLAY OR PLOT COMMAND N
     1           ,20HOT A NAME              )
10002 FORMAT(1H ,40HFIGURE NAME IN DISPLAY OR PLOT COMMAND N
     1           ,20HOT IN NAME TABLE       )
10003 FORMAT(1H ,40HNON-NUMERIC MANIPULATION CONSTANT IN PLO
     1           ,20HT OR DISPLAY COMMAND)
10004 FORMAT(1H ,20HSETUP ERROR IN WORD ,14,10H  OF LINE ,A6
     1           ,11H OF FIGURE ,A6         )
10005 FORMAT(1H ,40HNON-NUMERIC MANIPULATION CONSTANT IN SUB
     1           ,20HFIG LINE               )
10006 FORMAT(1H ,40HNON-NUMERIC PARAMETER IN DISPLAY OR PLOT
     1           ,20H COMMAND               )
10015 FORMAT(1H ,40HMORE THAN FOUR CARDS IN INPUT LINE       )
10016 FORMAT(1H ,40HILLEGAL KEY WORD IN INPUT LINE           )
10017 FORMAT(1H ,40HILLEGAL SYNTAX IN PRINT COMMAND          )
10018 FORMAT(1H ,40HARROW NOT ON FIGURE LINE IN PRINT FIGURE
     1           ,20H COMMAND               )
```

```
10019 FORMAT(1H ,40HARROW ON EOF LINE IN PRINT FIGURE COMMAN
     1           , 1HD)
10020 FORMAT(1H ,40HREACHED EOF LINE BEFORE N LINES PRINTED )
10021 FORMAT(1H ,40HNAME WORD IN FIGURE LINE NOT A NAME     )
10022 FORMAT(1H ,40HNAME IN FIGURE LINE ALREADY USED        )
10023 FORMAT(1H ,40HMAIN MEMORY ARRAY SIZE EXCEEDED         )
10024 FORMAT(1H ,40HINVALID KIND NUMBER IN B.C.U. LINE      )
10025 FORMAT(1H ,40HNAME WORD IN LOCATE COMMAND NOT A NAME  )
10026 FORMAT(1H ,40HNAME IN LOCATE COMMAND NOT DEFINED AS A
     1           ,20HFIGURE NAME          )
10027 FORMAT(1H ,40HEOF ENCOUNTERED BEFORE N LINES SKIPPED  )
10028 FORMAT(1H ,40HFIGURE LINE ENCOUNTERED BEFORE N LINES S
     1           , 6HKIPPED)
10029 FORMAT(1H ,40HFIRST LINE ENCOUNTERED BEFORE -N LINES S
     1           , 6HKIPPED)
10030 FORMAT(1H ,40HFIGURE LINE ENCOUNTERED BEFORE -N LINES
     1           , 7HSKIPPED)
10031 FORMAT(1H ,40HCOUNT WORD NEGATIVE IN DELETE COMMAND   )
10032 FORMAT(1H ,40HARROW NOT ON FIGURE LINE IN DELETE 0 COM
     1           , 4HMAND)
10033 FORMAT(1H ,40HCOUNT WORD INVALID IN DELETE COMMAND    )
10034 FORMAT(1H ,40HARROW ON EOF LINE IN DELETE 0 COMMAND   )
10035 FORMAT(1H ,40HFIGURE LINE OCCURS IN LESS THAN N LINES
     1           ,20HIN DELETING          )
10036 FORMAT(1H ,40HCOUNT WORD INVALID IN SKIP COMMAND      )
C
C
C       PLOTTER
C
C
C
C  PLOTTER PROGRAM INITIALIZATION.
C
 9000 K=0
      LPT1=0
      LPT2=0
      DO 9015 I=1,20
      IF(K) 9016,9017,9016
 9016 L(I)=6H
      GO TO 9015
 9017 N=IDENTY(LIN(I+9))
      IF(N) 9018,9012,9014
 9018 IF(N+2)9014,9013,9014
 9012 L(I)=LIN(I+9)
      GO TO 9015
 9013 K=1
      GO TO 9016
 9014 PRINT 10006
      GO TO 9999
 9015 CONTINUE
      CALL STACK2(0)
      DO 9005 I=6,9
      N=IDENTY(LIN(I))
```

```
      IF(N) 9006,9007,9006
 9007 LIN(I)=INTGER(LIN(I),N)
 9005 CONTINUE
      GO TO 9008
 9006 PRINT 10003
      GO TO 9999
 9008 A=1.0
      B=0.0
      C=0.0
      D=1.0
      S=0.0
      T=0.0
C
C  CALCULATE NEW A,B,C,D,S,T. FIND NEW NSI.
C
 9010 X=LIN(8)
      X=X/(1000*57.29578)
      X1=COS(X)
      X2=SIN(X)
      X3=LIN(9)
      X3=X3/1000.
      Y1=LIN(6)
      Y1=Y1/1000.
      Y2=LIN(7)
      Y2=Y2/1000.
      S=S+A*Y1+B*Y2
      T=T+C*Y1+D*Y2
      Y3=X3*(A*X1+B*X2)
      B=X3*(-A*X2+B*X1)
      A=Y3
      Y3=X3*(C*X1+D*X2)
      D=X3*(-C*X2+D*X1)
      C=Y3
      N=IDENTY(LIN(5))
      IF(N+1) 9002,9001,9002
 9002 PRINT 10001
      GO TO 9999
 9001 N=LSRCHN(LIN(5))
      IF(N) 9003,9003,9004
 9003 PRINT 10002
      GO TO 9999
 9004 NSI=N
      GO TO 9020
C
C  PLOT A FIGURE
C
 9020 NSI=L(NSI+1)+NSI
      IF(L(NSI)) 9050,9060,9021
 9021 IF(L(NSI)-10000) 9020,9020,9022
 9022 ID=1
      LT=1
C                        computed go to selection of program
 9023 N=L(NSI)/10000
```

```
      N=L(NSI)-N*10000
      NUM=N/1000
      KSN=(N-NUM*1000)/100
      GO TO (1000,2000,3000,4000,5000,6000,7000),NUM
 3000 PRINT  13000
13000 FORMAT(1H  ,20HROUTINE NUMBER ERROR)
      GO TO 9200
C
C  GO DOWN A STAGE
C
 9030 K=IGET(NSI,LT,N)
      IF(K) 9990,9990,9031
 9031 CALL STACK1(LT)
      CALL STACK1(NSI)
      ID=ID-1
      LT=1
      NSI=K
      GO TO 9023
C
C GO UP A STAGE.
C
 9040 ID=ID+1
      NSI=IPOP(1)
      LT=IPOP(1)+1
      GO TO 9023
C
C  GO DOWN A LEVEL.
C
 9050 K=A*1000.
      CALL STACK1(K)
      K=B*1000.
      CALL STACK1(K)
      K=C*1000.
      CALL STACK1(K)
      K=D*1000.
      CALL STACK1(K)
      K=S*1000.
      CALL STACK1(K)
      K=T*1000.
      CALL STACK1(K)
      CALL STACK1(NSI)
      CALL STACK2(NSI+5)
      DO 9051 I=1,4
      K=NSI+I+2
      N=IDENTY(L(K))
      IF(N) 9052,9051,9052
 9052 PRINT 10005
      GO TO 9999
 9051 LIN(I+5)=INTGER(L(K),N)
      LIN(5)=L(NSI+2)
      GO TO 9010
C
C GO UP A LEVEL.
```

```
C
 9060 N=IPOP(2)
      IF(N) 9062,9061,9062
 9061 GO TO 9999
C                  RECOPY AND EXIT FROM PLOTTER
 9062 NSI=IPOP(1)
      T=IPOP(1)
      T=T/1000.
      S=IPOP(1)
      S=S/1000.
      D=IPOP(1)
      D=D/1000.
      C=IPOP(1)
      C=C/1000.
      B=IPOP(1)
      B=B/1000.
      A=IPOP(1)
      A=A/1000.
      GO TO 9020
C
C  STAGE ERROR
C
 9990 LX=L(NSI-1)
 9993 IF(L(NSI)) 9991,9992,9991
 9991 NSI=NSI-L(NSI-2)
      GO TO 9993
 9992 LY=L(NSI-1)
      PRINT 10004, LT,LX,LY
      GO TO 9999
   99 GO TO 9990
C
C
C
C       B.C.U.  ANALYZING ROUTINES
C
C
C     CHOOSE ROUTINE FOR SPECIFIC POINT
C
 1000 GOTO (1100,1200,1300,1400),KSN
C
C
C     ROUTINE FOR POINT DEFINED BY TWO COORDINATES
C
 1100 GO TO (1110,1120,1130),LT
 1110 N=8
      K=IGET(NSI,LT,N)
      IF(K) 1111,9990,9031
 1111 MACC1=N
      LT=LT+1
      N=8
 1120 CALL STACK1(MACC1)
      N=8
      K=IGET(NSI,LT,N)
```

```
      IF(K) 1121,9990,9031
 1121 MACC1=N
      LT=LT+1
 1130 MACC2=MACC1
      MACC1=IPOP(1)
      GO TO 1050
C
C
C     ROUTINE FOR POINT DEFINED BY INTERSECTION OF TWO LINES
C
 1200 GO TO (1210,1220,1230),LT
 1210 N=1
      GO TO 9030
 1220 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 1230 X2=MACC1
      Y2=MACC2
      X1=IPOP(1)
      Y1=IPOP(1)
      DX1=X2-X1
      DX2=X4-X3
      IF(DX1) 1238,1234,1238
 1238 IF(DX2) 1231,1236,1231
 1231 XM1=(Y2-Y1)/DX1
      XM2=(Y4-Y3)/DX2
      DM=XM1-XM2
      IF(ABSF(DM)-1.) 1232,99,1232
 1232 MACC2=(Y3-XM2*X3+XM1*X1-Y1)/DM +0.5
      MACC1=(MACC2-Y3+XM2*X3)/XM2 +0.5
 1237 IF(ABSF(MACC2)-30000) 1233,99,99
 1233 IF(ABSF(MACC1)-30000)1050,99,99
 1234 IF(DX2) 1235,99,1235
 1235 MACC1=X1 +0.5
      MACC2=Y3+(Y4-Y3)/DX2*(X1-X3) +0.5
      GO TO 1237
 1236 MACC1=X3
      MACC2=Y1+(Y2-Y1)/DX1*(X3-X1) +0.5
      GO TO 1237
C
C
C     ROUTINE FOR POINT DEFINED BY TRANSLATION FROM A POINT
C
 1300 GOTO(1310,1320,1330,1340),LT
 1310 N=1
      GO TO 9030
 1320 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=8
      K=IGET(NSI,LT,N)
      IF(K) 1321,9990,9031
 1321 MACC1=N
```

```
      LT=LT+1
 1330 CALL STACK1(MACC1)
      N=8
      K=IGET(NSI,LT,N)
      IF(K) 1331,9990,9031
 1331 MACC1=N
      LT=LT+1
 1340 N=IPOP(1)+IPOP(1)
      MACC2=MACC1+IPOP(1)
      MACC1=N
      GO TO 1050
C
C
C     ROUTINE FOR PT DEFINED BY CIRCLE(ARC)
C
 1400 GO TO (1410,1420),LT
 1410 N=4
      GO TO 9030
 1420 GO TO 1050
C
C
C      ROUTINE TO DRAW A POINT
C
 1050 IF(ID-1) 9040,1051,9040
 1051 IF(LPLTSW) 1052,1052,1053
 1052 CONTINUE
C    ROUTINE TO DISPLAY A POINT
 1053 CONTINUE
C   ROUTINE TO PLOT A POINT
      GO TO 9020
C
C
C     CHOOSE ROUTINE FOR SPECIFIC LINE
C
 2000 GOTO (2100,2200),KSN
C
C
C     ROUTINE FOR LINE SEGMENT DEFINED BY TWO POINTS
C
 2100 GO TO (2110,2120,2130),LT
 2110 N=1
      GO TO 9030
 2120 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 2130 MACC3=MACC1
      MACC4=MACC2
      MACC1=IPOP(1)
      MACC2=IPOP(1)
      GO TO 2050
C
C
```

```
C     ROUTINE FOR LINE DEFINED BY POINT AND TRANSLATION
C
 2200 GO TO (2210,2220,2230,2240),LT
 2210 N=1
      GO TO 9030
 2220 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=8
      K=IGET(NSI,LT,N)
      IF(K) 2221,9990,9031
 2221 MACC1=N
      LT=LT+1
 2230 CALL STACK1(MACC1)
      N=8
      K=IGET(NSI,LT,N)
      IF(K) 2231,9990,9031
 2231 MACC1=N
      LT=LT+1
 2240 N=IPOP(1)+IPOP(1)
      MACC2=MACC1+IPOP(1)
      MACC1=MACC2
      GO TO 2050
C
C
C      DRAW A LINE SEGMENT
C
 2050 IF(ID-1) 9040,2051,9040
 2051 IF(LPLTSW) 2052,2052,2053
 2052 CONTINUE
C     ROUTINE TO DISPLAY A LINE SEGMENT
      GO TO 9020
 2053 CONTINUE
C     ROUTINE TO PLOT A LINE SEGMENT
      GO TO 9020
C
C
C     CHOOSE ROUTINE FOR SPECIFIC CIRCLE
C
 4000 GOTO (4100,4200,4300),KSN
C
C
C     ROUTINE FOR CIRCLE DEFINED BY PT AND DIST
C
 4100 GO TO (4110,4120,4130),LT
 4110 N=1
      GO TO 9030
 4120 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=6
      K=IGET(NSI,LT,N)
      IF(K) 4121,9990,9031
 4121 MACC1=N
      LT=LT+1
```

```
 4130 MACC3=MACC1
      MACC1=IPOP(1)
      MACC2=IPOP(1)
      GO TO 4050
C
C
C     ROUTINE FOR CIRCLE DEFINED BY CENTER AND PT ON CIRCUM
C
 4200 GO TO (4210,4220,4230),LT
 4210 N=1
      GO TO 9030
 4220 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 4230 X1=IPOP(1)
      Y1=IPOP(1)
      X2=MACC1
      Y2=MACC2
      MACC3=SQRT((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1))
      IF(MACC3-20) 99,99,4231
 4231 MACC1=X1
      MACC2=Y1
      GO TO 4050
C
C
C     ROUTINE FOR CIRCLE DEFINED BY 3 POINTS ON CIRCUM
C
 4300 GO TO (4310,4320,4330,4340),LT
 4310 N=1
      GO TO 9030
 4320 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 4330 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 4340 X3=MACC1
      Y3=MACC2
      X2=IPOP(1)
      Y2=IPOP(1)
      X1=IPOP(1)
      Y1=IPOP(1)
      C1=-(X1*X1+Y1*Y1)
      C2=-(X2*X2+Y2*Y2)
      C3=-(X3*X3+Y3*Y3)
      DET=X1*(Y2-Y3)-Y1*(X2-X3)+X2*Y3-X3*Y2
      IF(ABSF(DET)-5.)0099,4351,4351
 4351 D=(C1*(Y2-Y3)-Y1*(C2-C3)+C2*Y3-C3*Y2)/DET
      E=(X1*(C2-C3)-C1*(X2-X3)+X2*C3-X3*C2)/DET
      F=X1*(Y2*C3-C2*Y3)-Y1*(X2*C3-C2*X3)+C1*(X2*Y3-X3*Y2)
```

```
      F=F/DET
      MACC1=(-D/2.)
      MACC2=(-E/2.)
      MACC3=.5*SQRTF(D*D+E*E-4.*F)
      GO TO 4050
C
C
C     DRAW CIRCLE
C
 4050 IF(ID-1) 9040,4051,9040
 4051 IF(LPLTSW)4052,4052,4053
 4052 CONTINUE
C           ROUTINE TO DISPLAY A CIRCLE
      GO TO 9020
 4053 CONTINUE
C           ROUTINE TO PLOT A CIRCLE
      GO TO 9020
C
C
C     CHOOSE ROUTINE FOR SPECIFIC ARC
C
 5000 GOTO(5100,5200,5300,5400,5500,5600),KSN
C
C
C     ROUTINE FOR ARC DEFINED BY CIRCLE AND TWO ANGLES
C
 5100 GO TO (5110,5120,5130,5140),LT
 5110 N=4
      GO TO 9030
 5120 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=7
      GO TO 9030
 5130 CALL STACK1(MACC1)
      N=7
      GO TO 9030
 5140 IF(ABSF(MACC1)-3142) 5141,5141,5142
 5142 MACC1=MACC1-SIGNF(MACC1,3142)
 5141 MACC5=MACC1
      MACC1=IPOP(1)
      IF(ABSF(MACC1)-3142) 5143,5143,5144
 5144 MACC1=MACC1-SIGNF(MACC1,3142)
 5143 MACC4=MACC1
      MACC1=IPOP(1)
      MACC2=IPOP(1)
      MACC3=IPOP(1)
      GO TO 5050
C
C
C     ROUTINE FOR ARC DEFINED BY A CIRCLE AND TWO POINTS
C
 5200 GO TO (5210,5220,5230,5240),LT
```

```
 5210 N=4
      GO TO 9030
 5220 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 5230 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 5240 X3=MACC2
      Y3=MACC1
      X2=IPOP(1)
      Y2=IPOP(1)
      X1=IPOP(1)
      Y1=IPOP(1)
      PI=3141.59
      DY=Y2-Y1
      DX=X2-X1
      IF(DX) 5242,5241,5242
 5241 IF(DY) 5243,99,5244
 5243 ALP=PI/2.
      GO TO 5245
 5244 ALP=-PI/2.
      GO TO 5245
 5242 ALP=ATANF(DY/DX)*1000.
      IF(DX) 5245,5245,5246
 5246 ALP=ALP-SIGNF(PI,ALP)
 5245 MACC4=ALP
      DY=Y1-Y3
      DX=X1-X3
      IF(DX) 5262,5261,5262
 5261 IF(DY) 5263,99,5264
 5263 ALP=PI/2.
      GO TO 5265
 5264 ALP=-PI/2.
      GO TO 5265
 5262 ALP=ATANF(DY/DX)*1000.
      IF(DX) 5265,5265,5266
 5266 ALP=ALP-SIGNF(PI,ALP)
 5265 MACC5=ALP
      GO TO 5050
C
C
C     ROUTINE FOR ARC DETER BY CIRCLE,PT,ANGLE
C
 5300 GO TO (5310,5320,5330,5340),LT
 5310 N=4
      GO TO 9030
 5320 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
```

```
      N=1
      GO TO 9030
 5330 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=7
      GO TO 9030
 5340 A2=MACC1
      X1=IPOP(1)
      Y1=IPOP(1)
      X2=IPOP(1)
      Y2=IPOP(1)
      R=IPOP(1)
 5350 PI=3141.59
      DY=Y2-Y1
      DX=X2-X1
      IF(DX) 5342,5341,5342
 5341 IF(DY) 5343,99,5344
 5343 ALP=PI/2.
      GO TO 5345
 5344 ALP=-PI/2.
      GO TO 5345
 5342 ALP=ATANF(DY/DX)*1000.
      IF(DX) 5345,5345,5346
 5346 ALP=ALP-SIGNF(PI,ALP)
 5345 IF(A2) 5349,99,5347
 5349 A1=ALP+A2
      IF(ABSF(A1)-PI) 5351,5352,5352
 5352 A1=A1-SIGNF(2.*PI,A1)
 5351 A2=ALP
      GO TO 5348
 5347 A2=ALP+A2
      IF(ABSF(A1)-PI) 5353,5354,5354
 5354 A2=A2-SIGNF(2.*PI,A2)
 5353 A1=ALP
 5348 MACC1=X2
      MACC2=Y2
      MACC3=R
      MACC4=A1
      MACC5=A2
      GO TO 5050
C
C
C     ROUTINE FOR ARC DETER BY CIRCLE,PT,DIST
C
 5400 GO TO (5410,5420,5430,5440),LT
 5410 N=4
      GO TO 9030
 5420 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 5430 CALL STACK1(MACC2)
```

```
      CALL STACK1(MACC1)
      N=6
      K=IGET(NSI,LT,N)
      IF(K) 5431,9990,9031
 5431 MACC1=N
      LT=LT+1
 5440 D=MACC1
      X1=IPOP(1)
      Y1=IPOP(1)
      X2=IPOP(1)
      Y2=IPOP(1)
      R =IPOP(1)
 5441 IF(D-2.*R) 5442,5442,99
 5442 T=D/(2.*R)
      A2=2000.*ATANF(   T/SQRTF(1.-T*T))
      GO TO 5350
C
C
C     ROUTINE FOR ARC DETER BY CIRCLE,ANGLE DIST
C
 5500 GO TO (5510,5520,5530,5540),LT
 5510 N=4
      GO TO 9030
 5520 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 5530 CALL STACK1(MACC1)
      N=6
      K=IGET(NSI,LT,N)
      IF(K) 5531,9990,9031
 5531 MACC1=N
      LT=LT+1
 5540 D=MACC1
      A1=IPOP(1)
      X2=IPOP(1)
      Y2=IPOP(1)
      R =IPOP(1)
 5541 IF(D-2.*R) 5542,5542,99
 5542 T=D/(2.*R)
      A2=2000.*ATANF(   T/SQRTF(1.-T*T))
      GO TO 5351
C
C
C     ROUTINE FOR ARC DETER BY CIRCLE AND TWO DIST
C
 5600 GO TO (5610,5620,5630,5640),LT
 5610 N=4
      GO TO 9030
 5620 CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
```

```
      N=6
      K=IGET(NSI,LT,N)
      IF(K) 5621,9990,9031
 5621 MACC1=N
      LT=LT+1
 5630 CALL STACK1(MACC1)
      N=6
      K=IGET(NSI,LT,N)
      IF(K) 5631,9990,9031
 5631 MACC1=N
      LT=LT+1
 5640 D2=MACC1
      D1=IPOP(1)
      X1=IPOP(1)
      Y1=IPOP(1)
      R =IPOP(1)
 5641 IF(D1-2.*R) 5642,5642,99
 5642 T=D1/(2.*R)
      A1=2000.*ATANF(T/SQRTF(1.-T*T))
      IF(D2-2.*R) 5643,5643,99
 5643 T=D2/(2.*R)
      A2=2000.*ATANF(T/SQRTF(1.-T*T))
      MACC4=A1
      MACC5=A2
      MACC1=X1
      MACC2=X2
      MACC3=R
      GO TO 5050
C
C
C     DRAW AN ARC
C
 5050 IF(ID-1) 9040,5051,9040
 5051 IF(LPLTSW)5052,5052,5053
 5052 CONTINUE
C      ROUTINE TO DISPLAY ARC
      GO TO 9020
 5053 CONTINUE
C      ROUTINE TO PLOT ARC
      GO TO 9020
C
C
C     CHOOSE ROUTINE FOR SPECIFIC DISTANCE
C
 6000 GOTO (6100,6200,6300,6400,6500,6600,6700),KSN
C
C
C     ROUTINE FOR DISTANCE DEFINED BY TWO PTS
C
 6100 GOTO (6110,6120,6130),LT
 6110 N=1
      GO TO 9030
 6120 CALL STACK1(MACC1)
```

```
      CALL STACK1(MACC2)
      N=1
      GO TO 9030
 6130 X2=MACC1
      Y2=MACC2
      Y1=IPOP(1)
      X1=IPOP(1)
      MACC1=SQRTF((X2-X1)*(X2-X1)+(Y2-Y1)*(Y2-Y1))
      GO TO 6050
C
C
C     ROUTINE FOR HORZ DIST BETWEEN TWO PTS
C
 6200 GOTO (6210,6220,6230),LT
 6210 N=1
      GO TO 9030
 6220 CALL STACK1(MACC1)
      N=1
      GO TO 9030
 6230 MACC1=ABSF(MACC1-IPOP(1))
      GO TO 6050
C
C
C     ROUTINE FOR VERT DIST BETWEEN TWO PTS
C
 6300 GOTO (6310,6320,6330),LT
 6310 N=1
      GO TO 9030
 6320 CALL STACK1(MACC2)
      N=1
      GO TO 9030
 6330 MACC1= ABSF(MACC2-IPOP(1))
      GO TO 6050
C
C
C     ROUTINE FOR DIST DEFINED BY PT AND LINE
C
 6400 GOTO (6410,6420,6430),LT
 6410 N=1
      GO TO 9030
 6420 CALL STACK1(MACC1)
      CALL STACK1(MACC2)
      N=2
      GO TO 9030
 6430 X1=MACC1
      Y1=MACC2
      X2=MACC3
      Y2=MACC4
      Y3=IPOP(1)
      X3=IPOP(1)
      A=Y1-Y2
      B=X2-X1
      C=(Y2-Y1)*X1+(X1-X2)*Y1
```

```
      IF(ABSF(B)-5.)6433,6433,6434
 6433 MACC1=ABSF(X1-X3)
      GO TO 6050
 6434 IF(B)6431,6432,6432
 6432 MACC1=ABSF((A*X3+B*Y3+C)/SQRT(A*A+B*B))
      GO TO 6050
 6431 TX=X1
      TY=Y1
      X1=X2
      Y1=Y2
      X2=TX
      Y2=TY
      GOTO 6432
C
C
C     ROUTINE FOR DISTANCE DEFINED BY CIRCLE
C
 6500 GO TO (6510,6520),LT
 6510 N=4
      GO TO 9030
 6520 MACC1=MACC3
      GO TO 6050
C
C
C     ROUTINE FOR DIST -- REAL NUMBER
C
 6600 GO TO( 6610,6620),LT
 6610 N=8
      K=IGET(NSI,LT,N)
      IF(K) 6611,9990,9031
 6611 MACC1=N
      LT=LT+1
 6620 GO TO 6050
C
C
C     ROUTINE FOR BINARY OPERATION ON TWO DISTANCES
C
 6700 GO TO (6710,6720,6730,6740),LT
 6710 N=6
      K=IGET(NSI,LT,N)
      IF(K) 6711,9990,9031
 6711 MACC1=N
      LT=LT+1
 6720 CALL STACK1(MACC1)
      N=8
      K=IGET(NSI,LT,N)
      IF(K) 6721,9990,9031
 6721 MACC1=N
      LT=LT+1
 6730 CALL STACK1(MACC1)
      N=6
      K=IGET(NSI,LT,N)
      IF(K) 6731,9990,9031
```

```
 6731 MACC1=N
      LT=LT+1
 6740 N=IPOP(1)/1000
      A=IPOP(1)
      B=MACC1
      GOTO(6741,6742,6743,6744,6745,6746,6747),N
 6741 MACC1=A+B
      GO TO 6050
 6742 MACC1=A-B
      GO TO 6050
 6743 MACC1=A*B
      GO TO 6050
 6744 MACC1=A/B
      GO TO 6050
 6745 MACC1=A**B
      GO TO 6050
 6746 MACC1=A*SQRT(ABSF(B))
      GO TO 6050
 6747 MACC1=A*ABSF(B)
      GO TO 6050
C
C
C       DRAW A DISTANCE
C
 6050 GO TO 9040
C
C
C     CHOOSE ROUTINE FOR DEGREES
C
 7000 GOTO(7100,7200,7300),KSN
C
C
C     ROUTINE FOR DEGREES DETER BY TWO LINES
C
 7100 GO TO (7110,7120,7130),LT
 7110 N=2
      GO TO 9030
 7120 CALL STACK1(MACC4)
      CALL STACK1(MACC3)
      CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=2
      GO TO 9030
 7130 X3=MACC1
      Y3=MACC2
      X4=MACC3
      Y4=MACC4
      X1=IPOP(1)
      Y1=IPOP(1)
      X2=IPOP(1)
      Y2=IPOP(1)
 7141 IF(ABSF(X1-X2)-20.) 7143,7143,7144
 7143 A1=3.14159/2.
```

```
      GO TO 7142
 7144 A1=ATANF((Y1-Y2)/(X1-X2))
 7142 IF(ABSF(X3-X4)-20.) 7146,7146,7147
 7146 A2=3.14159/2.
      GO TO 7148
 7147 A2=ATANF((Y3-Y4)/(X3-X4))
 7148 MACC1=ABSF(A1-A2)
      GO TO 7050
C
C
C     ROUTINE FOR DEGREES DETER BY 3 PTS
C
 7200 GO TO (7210,7220,7230,7240),LT
 7210 N=1
      GO TO 9030
 7220 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 7230 CALL STACK1(MACC2)
      CALL STACK1(MACC1)
      N=1
      GO TO 9030
 7240 X3=MACC1
      Y3=MACC2
      X2=IPOP(1)
      Y2=IPOP(1)
      X1=IPOP(1)
      Y1=IPOP(1)
 7241 X4=X3
      Y4=Y3
      X3=X2
      Y3=Y2
      GO TO 7141
C
C
C           ANGLE DETERMINED BY A REAL NUMBER
C
 7300 GO TO (7310,7320),LT
 7310 N=8
      K=IGET(NSI,LT,N)
      IF(K) 7311,9990,9031
 7311 MACC1=N
      LT=LT+1
 7320 GO TO 7050
C
C
C       DRAW AN ANGLE
C
 7050 GO TO 9040
      END
```

```
      FUNCTION IDENTY(J)
C     INDENTIFIES TYPE OF WORD
      END


      FUNCTION IGET(I,J,N)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      NSI=I
      LT=J
      LP=LPT2
    3 K=NSI+LT+1
      M=IDENTY(L(K))
      IF(M-1) 2,1,2
C
C  PARAMETER
C
    1 M=INTGER(L(K),M)
      NSI=LST2(LP)-6
      LT=M+5
      LP=LP-1
      GO TO 3
C
    2 IF(M) 5,4,5
C
C  VALUE
C
    4 IF(N-8) 6,7,6
    6 IF(N-6) 15,7,15
   15 IGET=0
      RETURN
    7 N=INTGER(L(K),M)
      IGET=-1
      RETURN
C
    5 IF(M+1) 8,9,8
    8 IGET=0
      RETURN
C
C  NAME
C
    9 NSI=NSI-L(NSI-2)
      IF(L(NSI-1)-L(K)) 10,11,10
   10 IF(L(NSI)) 9,12,9
   12 IGET=0
      RETURN
   11 K=L(NSI)/10000
      K=(L(NSI)-K*10000)/1000
      IF(K-N) 13,14,13
   13 IGET=0
      RETURN
```

```
   14 IGET=NSI
      RETURN
      END


      FUNCTION INTGER(J,N)
C       CONVERTS WORD TO INTEGER VALUE
      END


      FUNCTION IPOP(M)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      IF(M-1) 1,1,2
    1 IPOP=LST1(LPT1)
      LPT1=LPT1-1
      RETURN
    2 IPOP=LST2(LPT2)
      LPT2=LPT2-1
      RETURN
      END


      FUNCTION   LALOCT(N,M)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      IF(M) 1,2,3
    2 RETURN
    1 IF(N+M-21) 4,4,5
    4 LALOCT=0
      RETURN
    5 N1=N-1
      LS=N
      LC=0
   18 LS=LS-L(LS-2)
      LC=LC+L(LS+1)
      IF(L(LS)) 15,16,15
   15 IF(M+LC) 18,17,17
   16 CALL NAMDEL(L(LS-1))
      GO TO 15
   17 DO 6 I=N1,LALOPT
    6 N2=I+M
      L(N2 )=L(I)
      LALOPT=LALOPT+M
      GO TO 9
    3 IF(LALOPT+M-1000) 7,7,4
    7 N1=LALOPT-N+2
      DO 8 I=1,N1
      N2=LALOPT-I+1
      LD=N2+M
    8 L(LD  )=L(N2)
      LALOPT=LALOPT+M
    9 IF(NTLPT) 10,10,11
```

```
   10 LALOCT=1
      RETURN
   11 DO 12 I=1,NTLPT
      IF(NTL(I,2)-N+1) 12,13,13
   13 NTL(I,2)=NTL(I,2)+M
   12 CONTINUE
      GO TO 10
      END


      FUNCTION LSRCHN(N)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      IF(NTLPT) 1,1,2
    1 LSRCHN=0
      RETURN
    2 DO 3 I=1,NTLPT
      IF(N-NTL(I,1)) 3,4,3
    3 CONTINUE
      GO TO 1
    4 LSRCHN=NTL(I,2)
      RETURN
      END


      SUBROUTINE MAN ( X,Y, X1,Y1)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      COMMON A,B,C,D,S,T
      X1=S+A*X+B*Y
      Y1=T+C*X+D*Y
      RETURN
      END


      SUBROUTINE NAMADD(N,M)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      IF(NTLPT-50) 1,2,2
    1 NTLPT=NTLPT+1
      NTL(NTLPT,1)=N
      NTL(NTLPT,2)=M
      RETURN
    2 PRINT  10014
      STOP 05
10014 FORMAT(1H ,40HFIGURE NAME TABLE SIZE EXCEEDED          )
      END


      SUBROUTINE NAMDEL(N)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      DO 1 I=1,NTLPT
```

```
      IF(NTL(I,1)-N) 1,2,1
    1 CONTINUE
      RETURN
    2 IF(I-NTLPT) 3,4,4
    3 NTL(I,1)=NTL(I+1,1)
      NTL(I,2)=NTL(I+1,1)
      I=I+1
      GO TO 2
    4 NTLPT=NTLPT-1
      RETURN
      END


      SUBROUTINE STACK1(K)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      LPT1=LPT1+1
      IF(LPT1-200) 2,2,1
    1 PRINT 10010
      STOP 01
    2 LST1(LPT1)=K
      RETURN
10010 FORMAT(1H ,40HSTACK 1 SIZE EXCEEDED                    )
      END


      SUBROUTINE STACK2(K)
      DIMENSION L(1000),LIN(29),LST1(200),LST2(10),NTL(50,2)
      COMMON  L,LIN,LST1,LST2,NTL,LPT1,LPT2,LALOPT,NTLPT
      LPT2=LPT2+1
      IF(LPT2-10) 2,2,1
    1 PRINT 10011
      STOP 02
    2 LST2(LPT2)=K
      RETURN
10011 FORMAT(1H ,40HSTACK 2 SIZE EXCEEDED                    )
      END
```

VI. SUGGESTIONS FOR FURTHER RESEARCH

During the creation of GDL-1 and GDL-2 certain decisions have been made to facilitate the initial implementation of the language. Some of these have been mentioned previously. In particular, the motivations for the creation of GDL-2 constituted a whole area of further research which was felt sufficiently important to be implemented in the original study.

The following areas are some of those which need further study.

1/. Dynamic storage allocation for stacks, tables, and block structure. Unsophisticated error routines currently simply state that space allotted has been exceeded, and terminate the program.

2/. In GDL-1, garbage collection routines for block structures. A record could be kept of the usage of each BCU block. When delete commands involve the removal of all usages of a BCU block, it should by physically removed, and its space re-allocated.

3/. x_0, y_0, θ, E could be allowed to be pointers, or parameters as well as values as is currently the case.

4/. Usage of secondary storage for subfigures, allowing the creation of a library of primary figures to be created, from which larger figures could be composed. Input - output routines for these would have to be created, and linkage systems defined and implemented.

5/. Default conditions should be specified for certain parameters (x_0, y_0, θ, E, for example). If in addition free format

for the input lines is implemented, the punctuation of this could allow omitted parameters.

6/. Abbreviated format for input lines, allowing more geometrical descriptions of figures. For example: A: L.S., THRU B, || C. might indicate that A is a line segment drawn through point B parallel to line C.

7/. Specification of whether or not particular parameters are to be manipulated or not. Currently parameters are handed down in absolute form and the figures resulting from their use then manipulated. It might prove useful also to allow the parameter to be inversely manipulated in the process of handing it down, so that when the figure was manipulated the parameter would appear in unmanipulated form. This could be done if A, B, C, D, S, T were stored in stack 2, so as to be available while chasing down parameters.

8/. A check could be made to prevent drawing BCU's which are too small to appear properly. Some record of the size of subfigure could be kept, and when calls are made to it, the size of it checked.

9/. A limiting subroutine could be incorporated to prevent the attempted drawing of BCU's which appear off the display screen or plotter. This would allow the detailed creation of a large diagram. If it were displayed in total the suggestion in 8 would prevent unnecessary detail. To study the detail, smaller sections could be displayed in blow-ups. The suggestion here would prevent trouble with those sections of the diagram appearing outside the 'frame' of the drawing.

10/. When using a CRT for interactive display, it might prove useful to have an automatic call to the display subroutine while creating the figure so as to allow immediate display of BCU's which are being created.

11/. The addition of more BCU's, including conic sections, and curve fitting routines.

12/. On an error condition, provision to continue with the drawing of those parts of the figure which are unaffected by the error. This should reduce the number of attempts necessary to correct the figure.

The following apply only to GDL-2:

13/. Extend LOCATE to X·Y, where X is a figure name, Y is a name of a line within a figure.

14/. While drawing, where a BCUL is based on no parameters, store its values to prevent having to recalculate it.

15/. It may prove useful to check at file creation time that names are not duplicated within a figure.

16/. Set up a dynamic name table, so that names used within a figure are not visible outside that figure.

17/. In the current GDL-2 it is not possible to base lines on lines from different figures, except through the use of parameters. This could be made possible with the proper stacking and use of A, B, C, D, S, T and free format forms like ABC·DEF·G where ABC is the name of a sub-figure line in the current figure, DEF is the name of a subfigure line in the figure called in ABC, G is a basic line in the figure called in DEF.

Generally it is felt that the data structures of GDL-2 are sufficiently superior, principally on account of their flexibility, to those of GDL-1 that further study of GDL-1 is not warranted.

VII. SUMMARY AND CONCLUSIONS

As has been discussed, the task of providing languages through which a computer can be used by those with little knowledge of the machines themselves is an important one.

In consideration of the facts that a system equipped with an x-y plotter and a console typewriter can be used as quite an accurate drafting tool, and that no language exists currently to utilize this fact, this work has studied some aspects of the data structures necessary to provide such a language. A tentative language giving the ability to describe and produce planar figures composed of straight line segments and circle arcs has been defined.

Two versions of this Graphics Display Language have been designed and implemented. These vary in the degree to which they are compiled into a condensed format at the time of entry into the GDL system. The first is based on a structure with blocks of storage assigned for each basic construction unit, figure, and set of parameter and manipulation data. All relational information between these blocks is embodied intrinsically in pointers from one to another. The second version of GDL is based on a file structure of lines of input information. Relational information between these lines is embodied in both their ordering within the file, and through names given to each line which are used for reference purposes in other lines.

The programs which handle the creation of the block and pointer structure for GDL-1 and the file structure for GDL-2 are referred to as the HANDLER in each case. These differ greatly

between the two versions of GDL, and the commands which can be given them reflect these differences.

The extents of these differences is increased by the desire to make the GDL system an interactive one in which it is possible to 'talk' between the computer and the user on-line while creating the block or file structure. Although this interactive nature is not intrinsic in the objective of this study as stated at the beginning of this summary it is an important aspect for the many time-sharing systems being designed. These systems are becoming more and more appreciated due to the powerful capabilities provided by the man-machine symbiosis they make possible. Therefore the system and structures have been designed to provide as much of this interactive capability as possible.

The displaying subprogram is essentially the same in these two versions. Two push-down stacks allow arbitrarily deep nesting of figures within figures, and arbitrarily deep dependence of basic construction units on others.

The structures designed and the restrictions placed upon the user through the definition of the Handler commands leave little doubt that the second version of the GDL system is a better structure.

Conceptually, as a file is an easily comprehended idea, the user has little more to learn than the language itself, whereas the block structure of GDL-1 requires more understanding of the implementation used.

The flexibility of the GDL-2 structure lends itself much more easily to the interactive environment of time-shared systems. Also it appears to provide the basis for expansion of the facilities

of GDL in many more directions and to a greater extent than GDL-1.

A further indication of the superiority of the structure of GDL-2 is the easier programming (in Fortran) that results from its implementation. The file system appears to be a more 'natural' structure for this language.

The value of the language itself will have to be determined through usage. It is anticipated that usage will also highlight those constructions most commonly used, thus providing a guide to useful shorthand notations and/or library subfigures.

It is expected that this study will lead to more sophisticated versions of GDL which will prove valuable additions to the library of problem-oriented languages.

REFERENCE

1. Sutherland, I. E., "Sketchpad: A Man-Machine Graphical Communication System," Defense Documentation Center, Technical Report No. 296, January, 1963.